

# PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD

Publication number: KR20050088995 (A)

Publication date: 2005-09-07

Inventor(s): RAPP JOHN W [US]; JACKSON LARRY [US]; JONES MARK [US]; CHERASARO TROY [US]

Applicant(s): LOCKHEED CORP [US]

Classification:

- international: G06F9/30; G06F9/38; G06F9/445; G06F9/46; G06F15/78; G06F9/30; G06F9/38; G06F9/445; G06F9/46; G06F15/76; (IPC1-7): G06F9/38; G06F15/78


- European: G06F9/3854


Application number: KR20057007750 20050430


Priority number(s): US20030683929 20031009; US20030683932 20031009; US20030684053 20031009; US20030684057 20031009; US20030684102 20031009; US20020422503P 20021031


Also published as:

 WO2004042560 (A2)

 WO2004042560 (A3)

 WO2004042574 (A2)

 WO2004042574 (A3)

 WO2004042569 (A2)

[more >>](#)

Abstract not available for KR 20050088995 (A)

Abstract of corresponding document: **WO 2004042560 (A2)**

A peer-vector machine includes a host processor and a hardwired pipeline accelerator. The host processor executes a program, and, in response to the program, generates host data, and the pipeline accelerator generates pipeline data from the host data. Alternatively, the pipeline accelerator generates the pipeline data, and the host processor generates the host data from the pipeline data. Because the peer-vector machine includes both a processor and a pipeline accelerator, it can often process data more efficiently than a machine that includes only processors or only accelerators. For example, one can design the peer-vector machine so that the host processor performs decision-making and non-mathematically intensive operations and the accelerator performs non-decision-making and mathematically intensive operations. By shifting the mathematically intensive operations to the accelerator, the peer-vector machine often can, for a given clock frequency, process data at a speed that surpasses the speed at which a processor-only machine can process the data.

.....  
Data supplied from the **esp@cenet** database --- Worldwide

## KOREAN PATENT ABSTRACTS

(11) Publication number: 1020050088995 A

(43) Date of publication of application: 07.09.2005

(21) Application number: 1020057007750

(22) Date of filing: 30.04.2005

(30) Priority: 2003 683929 US 09.10.2003

(51) Int. Cl: G06F 9/38 (2006.01);  
G06F 15/78 (2006.01);

(71) Applicant: LOCKHEED MARTIN CORPORATION

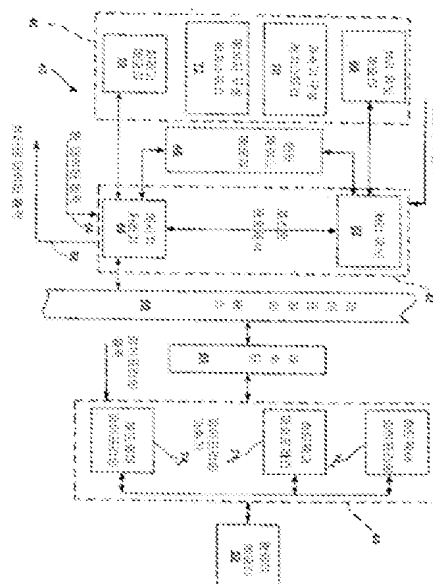
(72) Inventor: RAPP JOHN W.  
JACKSON LARRY  
JONES MARK  
CHERASARO TROY

## (54) PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD

## (57) Abstract:

A pipeline accelerator includes a memory and a hardwired-pipeline circuit coupled to the memory. The hardwired-pipeline circuit is operable to receive data, load the data into the memory, retrieve the data from the memory, process the retrieved data, and provide the processed data to an external source. In addition or in the alternative, the hardwired-pipeline circuit is operable to receive data, process the received data, load the processed data into the memory, retrieve the processed data from the memory, and provide the retrieved processed data to an external source. Where the pipeline accelerator is coupled to a processor as part of a peer-vector machine, the memory facilitates the transfer of data - whether unidirectional or bidirectional - between the hardwired-pipeline circuit(s) and an application that the processor executes.

© KIPO &amp; WIPO 2007



*This Facsimile First Page has been artificially created from the Korean Patent Abstracts CD Rom*

# (19)대한민국특허청(KR)

## (12) 공개특허공보(A)

(51) Int. Cl.<sup>7</sup>  
G06F 9/38  
G06F 15/78

(11) 공개번호 10-2005-0088995  
(43) 공개일자 2005년09월07일

(21) 출원번호 10-2005-7007750  
(22) 출원일자 2005년04월30일  
면역문 제출일자 2005년04월30일  
(86) 국제출원번호 PCT/US2003/034558  
국제출원일자 2003년10월31일

(87) 국제공개번호 WO 2004/042562  
국제공개일자 2004년05월21일

(30) 우선권주장 10/683,929 2003년10월09일 미국(US)  
10/683,932 2003년10월09일 미국(US)  
10/684,053 2003년10월09일 미국(US)  
10/684,057 2003년10월09일 미국(US)  
10/684,102 2003년10월09일 미국(US)  
60/422,503 2002년10월31일 미국(US)

(71) 출원인 록히드 마틴 코포레이션  
미국 버지니아주 20110, 마나사스, 글렌 드라이브 9500, 메일 드롭 043, 빌딩 400

(72) 발명자 램, 존 더블류  
미국 버지니아 20110, 매나사스, 리버 크레스트 로드 9350  
잭슨, 테리  
미국 버지니아 20112, 매나사스 크레스트록 드라이브 13093  
존, 마크  
미국 버지니아 20120, 샌트레발, 오크메이 플레이스 15342  
케라사로, 프로이  
미국 버지니아 22701, 켈페퍼, 캐스트랄 코트 1524

(74) 대리인 전영일  
염주석

심사청구 : 없음

### (54) 향상된 컴퓨팅 아키텍처를 위한 파이프라인 가속기 및 관련시스템 및 방법

#### 요약

파이프라인 가속기에는 메모리 및 상기 메모리와 결합된 하드와이어드-파이프라인 회로가 포함되어 있다. 상기 하드와이어드-파이프라인 회로는 데이터를 수신하고, 이 데이터를 메모리에 로드하고, 상기 메모리에서 이 데이터를 검색하고, 상기 검색된 데이터를 처리하고, 상기 처리된 데이터를 외부 소스로 제공하도록 동작 가능하다. 추가로 또는 선택적으로, 상기 하드와이어드-파이프라인 회로는 데이터를 수신하고, 상기 수신된 데이터를 처리하고, 상기 처리된 데이터를 메모리에 로드하고, 상기 메모리에서 상기 처리된 데이터를 검색하고, 상기 검색된 처리된 데이터를 외부 소스로 제공하도록 동

작 가능하다. 파이프라인 가속기가 피어-백터 머신의 일부로서의 프로세서와 결합하면, 상기 메모리는 상기 하드와이어드-파이프라인 회로와 상기 프로세서가 실행하는 애플리케이션 사이에서 ~ 단방향 또는 양방향으로 ~ 데이터 전송을 용이하게 한다.

대표도

도 3

색인어

파이프라인, 파이프라인 가속기, 하드와이어드-파이프라인

명세서

기술분야

(우선권 주장)

본 출원은 참고문헌으로서 통합되는 2002년 10월 31일 출원된 미국 가출원 제60/422,503호의 우선권을 주장한다.

(관련된 출원과의 상호참조)

본 출원은 발명의 명칭이 "향상된 컴퓨팅 아키텍처 및 관련 시스템 및 방법" 인 미국 특허출원 제10/684,102호, 발명의 명칭이 "향상된 컴퓨팅 아키텍처를 가지는 컴퓨팅 머신 및 관련 시스템 및 방법" 인 미국 특허출원 제 10/684,053호, 발명의 명칭이 "프로그램가능한 회로 및 관련 컴퓨팅 머신 및 방법" 인 미국 특허출원 제10/684,057호, 및 발명의 명칭이 "다중 파이프라인 유닛을 가지는 파이프라인 가속기 및 관련 컴퓨팅 머신 및 방법" 인 미국 특허출원 제 10/684,932호와 관련이 되어 있으며, 상기 미국 특허출원 발명은 모두 2003년 10월 9일 동일한 권리자에 의해 출원되었으며, 본 명세서에 참고문헌으로 통합된다.

배경기술

상대적으로 짧은 시간에서 상대적으로 대용량의 데이터를 처리하기 위한 일반적인 컴퓨팅 아키텍처(computing architecture)에는 부하를 분배하는 다중 상호접속 프로세서(multiple interconnected processor)가 포함되어 있다. 처리 부하를 분배함으로써, 이들 다중 프로세서들은 주어진 클럭 주파수에서 하나의 프로세서가 할 수 있는 것 보다 더욱 빨리 데이터를 처리할 수 있다. 예를 들어, 프로세서 각각은 데이터의 일부를 처리 하거나 또는 처리되는 알고리즘 일부를 실행할 수 있다.

도 1은 다중-프로세서 아키텍처를 갖는 종래의 컴퓨팅 머신(computing machine)(10)의 개략적인 블록 다이어그램이다. 머신(10)에는 마스터 프로세서(12) 및 버스(16)를 통해 상기 마스터 프로세서와 상호 통신을 하는 코프로세서( $14_1$ - $14_n$ ), 원격 장치(도 1에는 도시하지 않음)로부터 원 데이터(raw data)를 수신하는 입력 포트(18), 및 처리된 데이터를 상기 원격 소스로 제공하는 출력 포트(20)가 포함되어 있다. 상기 머신(10)에는 또한 마스터 프로세서(12)용 메모리(22), 코프로세서( $14_1$ - $14_n$ )용 메모리( $24_1$ - $24_n$ ), 및 상기 버스(16)를 통해 마스터 프로세서와 코프로세서가 공유하는 메모리(26)도 포함되어 있다. 메모리(22)는 마스터 프로세서(12)를 위한 프로그램 및 작업 메모리 모두의 역할을 하고, 메모리( $24_1$ - $24_n$ ) 각각은 코프로세서( $14_1$ - $14_n$ ) 각각을 위한 프로그램 및 작업 메모리 모두의 역할을 한다. 공유 메모리(26)는 마스터 프로세서(12)와 코프로세서(14)가 그들 사이의 데이터를 포트(18 및 20)를 통해 각각 원격 장치로/장치로부터 전달할 수 있게 해준다. 마스터 프로세서(12) 및 코프로세서(14)는 또한 머신(10)이 원 데이터를 처리하는 속도를 제어하는 공통의 클럭 신호를 수신하기도 한다.

일반적으로, 컴퓨팅 머신(10)은 마스터 프로세서(12)와 코프로세서(14) 간의 원 데이터 처리를 효과적으로 분배한다. 소나 어레이(sonar array)와 같은 원격 소스(도 1에는 도시하지 않음)는 포트(18)를 통해 원 데이터를 상기 원 데이터를 위한 선입선출(FIFO) 버퍼(도시하지 않음)로서 작동하는 공유 메모리(26)의 일부에 로드한다. 마스터 프로세서(12)는 버스(16)를 통해 메모리(16)로부터 원 데이터를 검색하고, 마스터 프로세서와 코프로세서(14)가 상기 원 데이터를 처리하고,

버스(16)를 통해 필요한 만큼 그들사이에서 데이터를 전달한다. 마스터 프로세서(12)는 처리된 데이터를 공유 메모리(26)에 정의되어 있는 다른 FIFO 버퍼(도시하지 않음)에 로드하고, 원격 소스가 포트(20)를 통해 이 FIFO로부터 상기 처리된 데이터를 검색한다.

다른 연산의 예에서, 컴퓨팅 머신(10)은 원 데이터를 처리하는데 있어서 상기 원 데이터상에서 각각의 연산에  $n+1$ 을 순차적으로 수행하여 처리하는데, 이 연산은 패스트 푸리에 변환(FFT)과 같은 처리 알고리즘을 함께 구성한다. 보다 특별하게는, 머신(10)은 마스터 프로세서(12)와 코프로세서(14)로부터 데이터-처리 파이프라인을 형성한다. 주어진 블록 신호의 주파수를 위해, 그러한 파이프라인은 종종 머신(10)이 하나의 프로세서만 가지는 머신보다 더욱 빠르게 원 데이터를 처리할 수 있게 한다.

메모리(26)내의 원-데이터 FIFO(도시하지 않음)로부터 원 데이터를 검색한 후, 마스터 프로세서(12)는 삼각 함수와 같은 제1 연산을 상기 원 데이터상에 수행한다. 이 연산은 프로세서(12)가 메모리(26) 내부에 정의된 제1-결과 FIFO(도시하지 않음)내에 저장하는 첫번째 결과를 산출해 낸다. 일반적으로, 프로세서(12)는 메모리(12) 내에 저장된 프로그램을 수행하며, 그 프로그램의 제어하에 상기 설명된 연산들을 수행한다. 프로세서(12)는 또한 메모리(22)를 작업 메모리로 사용하여 프로세서가 상기 제1 연산의 중간 시간에 발생하는 데이터를 일시적으로 저장하기도 한다.

다음으로, 상기 메모리(26)내의 제1-결과 FIFO(도시하지 않음)로부터 첫번째 결과를 검색한 후, 코프로세서(14)는 로그 함수와 같은 두번째 연산을 상기 첫번째 결과상에 수행한다. 이 두번째 연산은 코프로세서(14<sub>1</sub>)가 메모리(26) 내부에 정의된 제2-결과 FIFO(도시하지 않음)내에 저장하는 두번째 결과를 산출해 낸다. 일반적으로, 코프로세서(14<sub>1</sub>)는 메모리(24<sub>1</sub>)내에 저장된 프로그램을 수행하며, 그 프로그램의 제어하에 상기 설명된 연산들을 수행한다. 코프로세서(14)는 또한 메모리(24<sub>1</sub>)를 작업 메모리로 사용하여 코프로세서가 상기 제2 연산의 중간 시간에 발생하는 데이터를 일시적으로 저장하기도 한다.

그리고 나서, 코프로세서(24<sub>2</sub>-24<sub>n</sub>)는 상기 두번째-( $n-1$ )번째 상에 세번째-n번째 연산을 순차적으로 수행하여 상기 코프로세서(24<sub>1</sub>)를 위해 상기 설명한 것과 유사한 방법의 결과를 가져온다.

코프로세서(24<sub>n</sub>)에 의해 수행되는  $n$ 번째 연산은 최종 결과, 즉 처리된 데이터를 산출한다. 코프로세서(24<sub>n</sub>)는 메모리(26) 내부에 정의된 처리된-데이터 FIFO(도시하지 않음)로 이 처리된 데이터를 로드(load)하고, 원격 장치(도 1에는 도시하지 않음)가 이 FIFO로부터 상기 처리된 데이터를 검색한다.

마스터 프로세서(12)와 코프로세서(14)가 처리 알고리즘의 다른 연산을 동시에 수행하기 때문에, 컴퓨팅 머신(10)은 서로 다른 연산을 순차적으로 수행하는 하나의 프로세서를 갖는 컴퓨팅 머신 보다도 원 데이터를 보다 빨리 처리할 수 있기도 하다. 특히, 하나의 프로세서는 원 데이터의 앞 세트상에 모든  $n+1$  연산을 수행할 때 까지는 원 데이터의 새로운 세트를 검색할 수 없다. 그러나, 상기 언급한 파이프라인 기술을 이용해서, 마스터 프로세서(12)는 오직 첫번째 연산을 수행한 후 원 데이터의 새로운 세트를 검색할 수 있다. 따라서, 주어진 블록 주파수를 위해, 이 파이프라인 기술은 머신(10)이 원 데이터를 처리하는데 있어서 하나의 프로세서 머신(도 1에는 도시하지 않음)과 비교할 때 대략  $n+1$  배 더 빠른 원 데이터를 처리하는 속도를 증가시킬 수 있다.

선택적으로, 컴퓨팅 머신(10)은 원 데이터상에, FFT와 같은 처리 알고리즘의 경우에  $n+1$ 을 동시에 수행함으로써 병렬로 원 데이터를 처리할 수도 있다. 즉, 알고리즘에 앞서의 실시예에서 상기 언급한 바와 같은  $n+1$  순차적 연산이 포함되어 있다면, 마스터 프로세서(12)와 코프로세서(14) 각각은 모든  $n+1$  연산을 원 데이터 각각의 세트상에서 수행한다. 따라서, 주어진 블록 주파수를 위해서는, 상기 설명한 파이프라인 기술과 같이, 이러한 병렬-처리 기술은 머신(10)이 하나의 프로세서 머신(도 1에는 도시하지 않음)과 비교할 때 대략  $n+1$  배 더 빠른 원 데이터 처리 속도를 증가시킬 수 있다.

불행하게도, 비록 컴퓨팅 머신(10)이 하나의 프로세서 컴퓨터 머신(도 1에는 도시하지 않음)보다 빨리 데이터를 처리할 수는 있으나, 머신(10)의 데이터-처리 속도가 종종 프로세서 블록의 주파수보다 적어지곤 한다. 특히, 컴퓨팅 머신(10)의 데이터-처리 속도는 마스터 프로세서(12)와 코프로세서(14)가 데이터를 처리하는데 요구하는 시간에 의해 제한된다. 간략히 하기 위해, 이 속도 제한의 한 예를 마스터 프로세서(12)를 참고하여 설명하는데, 이 설명은 코프로세서(14)에도 적용됨을 이해할 수 있을 것이다. 앞에서 설명한 바와 같이, 마스터 프로세서(12)는 프로세서를 제어하여 데이터를 원하는 방식으로 조작하도록 제어하는 프로그램을 실행한다. 이 프로그램에는 프로세서(12)가 실행하는 일련의 명령이 포함되어 있다. 불행하게도, 프로세서(12)는 일반적으로 하나의 명령을 수행하는데 다중 블록 사이클을 필요로 하는데, 종종 다중 명령을 수행하여 데이터의 하나의 값을 처리해야 한다. 예를 들어, 프로세서(12)가 제1 데이터 값(A)(도시하지 않음)과 제2

데이터 값(B)(도시하지 않음)을 곱하는 경우를 가정한다. 제1 클럭 사이클 동안, 프로세서(12)는 메모리(22)로부터 여러개의 명령을 검색한다. 제2 및 제3 클럭 사이클 동안, 프로세서(12)는 메모리(22)로부터 A와 B를 각각 검색한다. 제4 클럭 사이클 동안, 프로세서(12)는 A와 B를 곱하고, 제5 클럭 사이클 동안, 그 결과물을 메모리(22 또는 26)에 저장하거나 또는 그 결과물을 원격 장치(도시하지 않음)로 제공한다. 이것은 최선의 시나리오인데, 그 이유는 많은 경우에서, 프로세서(12)는 카운터를 초기화(initializing) 및 닫는(closing) 경우와 같이 오버헤드 태스크(overhead task)를 위한 추가의 클럭 사이클을 요구하기 때문이다. 그러므로, 프로세서(12)는 A와 B를 처리하기 위해서는 5개의 클럭 사이클, 또는 데이터 값 당 평균 2.5개의 클럭 사이클을 필요로 한다.

따라서, 컴퓨팅 머신(10)이 데이터를 처리하는 속도는 종종 마스터 프로세서(12) 및 코프로세서(14)를 구동하는 클럭의 주파수보다 크게 낮아지곤 한다. 예를 들어, 프로세서(12)가 1.0 기가헤르츠(GHz)에서 클럭되지만 데이터 값 당 평균 2.5 클럭 사이클을 요구한다면, 유효 데이터-처리 속도는  $(1.0\text{GHz})/2.5=400\text{MHz}$  가 될 것이다. 이 유효 데이터-처리 속도는 종종 초당 연산 단위로 측정되곤 한다. 그러므로, 1.0GHz 의 클럭 속도를 위해서는, 프로세서(12)는 초당 0.4 기가연산(Gops)의 데이터-처리 속도로 레이트(rate)되어야 한다.

도 2는 프로세서가 주어진 클럭 주파수 및 종종 파이프라인이 클럭되는 레이트의 거의 동일한 레이트에서 할 수 있는 것 보다 더 빠른 일반적인 데이터를 처리할 수 있는 하드와이어드(hardwired) 데이터 파이프라인(30)의 블록 다이어그램이다. 파이프라인(30)에는 각각 실행 프로그램 명령 없이 각각의 데이터상의 개별적 연산을 각각 수행하는 연산자 회로(32<sub>1</sub>~32<sub>n</sub>)가 포함되어 있다. 즉, 원하는 연산이 회로(32)로 "번 들어(burned in)" 것으로 프로그램 명령 없이도 자동적으로 연산을 실행하는 것이다. 실행 프로그램 명령과 관련된 오버헤드를 제어함으로써, 파이프라인(30)은 종종 주어진 클럭 주파수를 위해 할 수 있는 것 보다 더 많은 연산을 수행할 수 있다.

예를 들어, 파이프라인(30)은 프로세서가 주어진 클럭 주파수를 위해 할 수 있는 것 보다 더 빠른 다음과 같은 수식을 풀 수 있다.

$$Y(X_k)=(5X_k+3)2^{xk}$$

여기서,  $X_k$ 는 일련의 원 데이터 값을 나타낸다. 이 예에서, 연산자 회로(32<sub>1</sub>)는  $5X_k$  를 계산하는 곱셈기이고, 회로(32<sub>2</sub>)는  $5X_k + 3$  을 계산하는 가산기이며, 회로(32<sub>n</sub>)( $n=3$ )는  $(5X_k + 3)2^{xk}$  를 계산하는 곱셈기이다.

제1 클럭 사이클  $k=1$  동안, 회로(32<sub>1</sub>)는 데이터 값( $X_1$ )을 수신하고 여기에 5를 곱해  $5X_1$  을 발생한다.

제2 클럭 사이클  $k=2$  동안, 회로(32<sub>2</sub>)는 회로(32<sub>1</sub>)로부터  $5X_1$  을 수신하고, 3을 더해서  $5X_1 + 3$  을 발생한다. 또한, 상기 제2 클럭 사이클 동안, 회로(32<sub>1</sub>)는  $5X_2$  를 발생한다.

제3 클럭 사이클  $k=3$  동안, 회로(32<sub>3</sub>)는 회로(32<sub>2</sub>)로부터  $5X_1 + 3$  을 수신하고,  $2^{x1}$  ( $x1$  만큼 유효하게  $5X_1 + 3$  왼쪽 시프트)를 곱하여 첫번째 결과( $5X_1 + 3$ ) $2^{x1}$  을 발생한다. 또한, 제3 클럭 사이클 동안, 회로(32<sub>1</sub>)는  $5X_3$  를 발생하고 회로(32<sub>2</sub>)는  $5X_2 + 3$  을 발생한다.

파이프라인(30)은 이러한 방식으로 모든 원 데이터 값이 처리될 때 까지 연속적인 원 데이터 값( $X_k$ )을 계속 처리한다.

따라서, 원 데이터 값( $X_1$ )을 수신한 후 두 개의 클럭 사이클의 지연 - 이 지연을 파이프라인(30)의 레이턴시(latency)라고 부르는 항 - 상기 파이프라인은 결과  $(5X_1 + 3)2^{x1}$  을 발생하고, 그 후에 하나의 결과 - 예를 들어,  $(5X_2 + 3)2^{x2}$ ,  $(5X_3 + 3)2^{x3}$ , ...,  $(5X_n + 3)2^{xn}$ , 를 각각의 클럭 사이클을 발생한다.

상기 레이턴시를 무시하면, 파이프라인(30)은 할록 속도와 동일한 데이터-처리 속도를 갖는다. 비교를 하면, 프로세서(12)와 코프로세서(14)가 상기 예에서와 같은 할록 속도의 0.4배의 데이터-처리 속도를 갖는다고 가정하면, 파이프라인(30)은 주어진 할록 속도를 위해 컴퓨팅 머신(10)(도 1)보다 2.5배 빨리 데이터를 처리할 수 있다.

도 2를 계속 참조하면, 설계자는 파이프라인(30)을 펠드-프로그래밍가능한 게이트 어레이(FPGA)와 같은 프로그래밍가능한 로직 IC(PLIC)에서 수행하도록 선택하기도 하는데, 그 이유는 PLIC는 주문형 반도체(ASIC)보다 나은 디자인 및 변형 유연성 가진다. PLIC 내부에서 하드와이어드 접속을 구성하기 위해서는, 설계자는 단지 PLIC 내부에 배치된 상호접속-구조 레지스터를 미리 결정된 이진 상태(binary state)로 설정하기만 하면 된다. 이들 이진 상태 모두의 조합을 "펌웨어(firmware)"라고 부르는 한다. 일반적으로, 설계자는 이 펌웨어를 PLIC 와 결합된 비휘발성 메모리(도 2에 도시하지 않음)에 로드한다. 누군가가 PLIC 를 "켜면(turn on)", PLIC는 상기 메모리로부터 펌웨어를 상기 상호접속-구조 레지스터로 다운로드한다. 그러므로, PLIC 의 기능을 변경시키기 위해서는, 설계자는 단지 펌웨어를 수정하면 되고 PLIC 로 하여금 그 수정된 펌웨어를 상호접속-구조 레지스터로 다운로드하게 하면 된다. 이것은 단지 펌웨어를 수정하는 것에 의해 PLIC를 수정할 수 있다는 것은 시계통화 단계 동안 및 "펠드 내에서" 파이프라인(30)의 업그레이드를 위해 특히 유용하다.

불행하게도, 하드와이어드 파이프라인(30)은 중요한 결정 형성, 특히 내포된 결정 형성(nested decision making)을 필요로 하는 알고리즘을 실행하는데 적선의 선택이 되지 못한다. 프로세서는 비교가능한 길이의 연산 명령(예를 들어, "A+B")을 실행할 수 있는 정도로 거의 빠르게 일반적으로 내포된-결정-형성 명령(예를 들어, "A이면 B를 행하고, 그렇지 않고 C이면 D를 행하고, ... 그렇지 않으면 n" 과 같은 내포된 조건 명령)를 실행할 수 있다. 그러나, 비록 파이프라인(30)이 상대적으로 간단한 결정(예를 들어, "A>B?")을 유효하게 구성할 수 있어도, 일반적으로 내포된 결정(예를 들어, "A이면 B를 행하고, 그렇지 않고 C이면 D를 행하고, ... 그렇지 않고 n")을 프로세서가 할 수 있는 것 만큼 유효하게 실행할 수 없다. 이러한 비효율성의 한 이유는 파이프라인(30)은 온-보드 메모리(on-board memory)가 거의 없어서 외부 작업/명령 메모리(도시하지 않음)를 액세스 할 필요가 있기 때문이다. 그리고, 비록 그러한 내포된 결정을 실행하기 위해 파이프라인(30)을 디자인할 수 있다 하여도, 요구되는 회로의 크기와 복잡성은 종종 설계를 불가능하게 만드는데, 특히 여러개의 서로 다른 내포된 결정을 포함하는 알고리즘인 경우 그러하다.

따라서, 프로세서는 중요한 결정 형성을 요구하는 애플리케이션 내에서 종종 사용되며, 하드와이어드 파이프라인은 결정 형성이 거의 없는 또는 전혀 없는 "수치치리(number crunching)" 애플리케이션으로 제한되곤 한다.

더욱이, 아래에 설명한 바와 같이, 도 2의 파이프라인(30)과 같은 하드와이어드 파이프라인, 특히 여러개의 PLIC를 포함하는 파이프라인(30)을 설계/수정하는 것 보다는 도 1의 컴퓨팅 머신(10)과 같은 프로세서-기반 컴퓨팅 머신을 설계/수정하는 것이 훨씬 쉽다.

프로세서와 그 주변장치들(예를 들어, 메모리)과 같은 컴퓨팅 구성요소들은 일반적으로 이 구성요소들의 상호접속을 촉진하여 프로세서-기반 컴퓨팅 머신을 형성하기 위한 산업-표준 통신 인터페이스를 포함한다.

특히, 표준 통신 인터페이스에는 두 개의 계층(layer)이 포함되는데, 물리 계층과 서비스 계층이다. 물리 계층에는 회로 및 이 회로의 연산 파라메터 및 인터페이스를 형성하는 대응 회로 상호접속이 포함되어 있다. 예를 들어, 물리 계층에는 구성요소를 버스와 연결하는 편, 이 편으로부터 데이터를 래치(latch)하는 버퍼, 및 이 편상에서 신호를 구동시키는 구동기가 포함되어 있다. 상기 연산 파라메터에는 상기 편이 수신하는 데이터 신호의 허용가능한 전압 범위, 데이터를 기록 및 관독하는 신호 타이밍, 및 지지된 연산 모드(예를 들어, 버스트 모드, 페이지 모드)가 포함되어 있다. 종래의 물리 계층에는 트랜지스터-트랜지스터 논리(TTL) 및 램버스(RAMBUS)가 포함된다.

서비스 계층에는 컴퓨팅 구성요소가 데이터를 전송하는 프로토콜이 포함된다. 이 프로토콜은 데이터의 포맷 및 상기 구성요소가 포맷된 데이터를 송수신하는 방식을 정의한다. 종래의 통신 프로토콜에는 파일-전송 프로토콜(FTP) 및 전송 제어 프로토콜/인터넷 프로토콜(TCP/IP)이 포함된다.

따라서, 산업-표준 통신 인터페이스를 갖는 제조자 및 다른 일반적인 설계 컴퓨팅 구성요소들로 인해서, 그러한 구성요소의 인터페이스를 일반적인 설계로 할 수 있고 이것을 상대적으로 적은 노력으로 다른 컴퓨팅 구성요소들과 상호접속시킬 수 있다. 이것은 설계자에게 대부분의 시간을 컴퓨팅 머신의 다른 부분들을 설계하는데 소비하게 만들고, 구성요소들을 추가하거나 없애는 것을 통해 머신을 쉽게 수정할 수 있게 한다.

산업-표준 통신 인터페이스를 지원하는 컴퓨팅 구성요소를 설계하는 것은 설계 라이브러리(design library)로부터 형편하는 물리-계층 설계를 사용하여 설계 시간을 절약하게 해 준다. 이것은 또한 구성요소들은 재고품인 컴퓨팅 구성요소들과 쉽게 접속할 수 있게 해주기도 한다.

공통의 산업-표준 통신 인터페이스를 지원하는 컴퓨팅 구성요소를 사용하여 컴퓨팅 머신을 설계하는 것은 설계자로 하여금 시간과 노력을 줄여주면서 구성요소들을 상호접속할 수 있게 해 준다. 이들 구성요소들이 공통의 인터페이스를 지원하기 때문에, 설계자는 설계 노력을 거의 들이지 않고 시스템 비스를 통해 이들을 상호 접속할 수 있다. 지원되는 인터페이스가 산업 표준이기 때문에, 설계자는 머신을 쉽게 수정할 수 있다. 예를 들어, 설계자는 다른 구성요소 및 주변장치들을 머신에 추가하여 시스템 설계를 발전시켜 나갈 수 있으며, 또는 차세대 구성요소들을 쉽게 추가/설계하여 기술 발전을 이룰 수 있다. 더욱이, 구성요소들이 공통의 산업-표준 서비스 계층을 지원하기 때문에, 설계자는 컴퓨팅 머신의 소프트웨어로 대응하는 프로토콜을 실현하는 현존하는 소프트웨어 모듈을 합체시킬 수 있다. 따라서, 설계자는 인터페이스 설계가 이미 적절하게 될수적이기 때문에 거의 노력을 들이지 않고 구성요소들을 접속시킬 수 있어서 머신이 특정 기능을 수행하게 하는 머신의 일부(예를 들어, 소프트웨어)를 설계하는 데 주력할 수 있다.

그러나, 불행하게도, 도 2의 파이프라인(30)과 같은 하드와이어드 파이프라인을 형성하는데 사용되는 PLIC 등과 같은 구성요소를 위한 알려진 산업-표준 서비스 계층은 없다.

따라서, 여러개의 PLIC 를 갖는 파이프라인을 설계하기 위해서, "스크래치(scratch)로부터" PLIC 간의 통신 인터페이스의 서비스 계층을 설계하고 디버깅하는데 상당한 시간과 노력을 들여야 한다. 일반적으로, ad hoc 서비스 계층은 PLIC 간에서 전달되는 데이터의 파라미터에 따라 달라진다. 비슷하게, 프로세서와 접속되는 파이프라인을 설계하기 위해서는, 설계자는 스크래치로부터 파이프라인과 프로세서간의 통신 인터페이스의 서비스 계층을 설계하고 디버깅하는데 상당한 시간과 노력을 들여야 한다.

비슷하게, PLIC 을 추가하는 것으로 파이프라인을 수정하기 위해서는, 설계자는 일반적으로 추가된 PLIC와 현재의 PLIC 사이의 통신 인터페이스의 서비스 계층을 설계하고 디버깅하는데 상당한 시간과 노력을 들이게 된다. 그리고, 프로세서를 추가하는 것으로 파이프라인을 수정하려면, 또는, 파이프라인을 추가하는 것으로 컴퓨팅 머신을 수정하려면, 설계자는 파이프라인과 프로세서간의 통신 인터페이스의 서비스 계층을 설계하고 디버깅하는데 상당한 시간과 노력을 들여야 한다.

그러므로, 도 1 및 도 2를 참고하면, 다수의 PLIC 를 접속하고 파이프라인과 프로세서의 접속 어려움으로 인해, 설계자는 컴퓨팅 머신을 설계하는 경우 상당한 트레이드오프(tradeoff)에 직면하곤 한다. 예를 들어, 프로세서-기반 컴퓨팅 머신을 가지고는, 설계자는 복잡한 결정-형성 가능성을 위한 트레이드 수-크런칭 속도 및 설계/수정 유연성에 집중하게 된다. 반대로, 하드와이어드 파이프라인-기반 컴퓨팅 머신을 가지고는, 설계자는 수-크런칭 속도를 위한 트레이드 복잡성-결정-형성 가능성 및 설계/수정에 집중하게 된다. 더욱이, 다수의 PLIC 를 접속하는 데의 어려움으로 인해, 소수의 PLIC 를 가지는 파이프라인-기반 머신을 설계하는 것이 불가능하기도 하다. 그 결과, 설계적인 파이프라인-기반 머신은 제한된 기능을 갖춘 한다. 그리고, 프로세서와 PLIC 와의 접속 어려움으로 인해서, 하나의 PLIC 이상과 프로세서와의 접속이 불가능하기도 하다. 그 결과, 프로세서와 파이프라인을 합침으로써 얻어지는 이익이 적다.

따라서, 하드와이어드-파이프라인-기반 머신의 수-크런칭 속도를 가지고 프로세서-기반 머신의 결정-형성 가능성을 결합시킬 수 있는 새로운 컴퓨팅 아키텍처 요구가 있어 왔다.

#### 발명의 상세한 설명

##### (요약)

본 발명의 한 실시예에 따르면, 파이프라인 가속기에는 메모리 및 메모리와 결합된 하드와이어드-파이프라인 회로가 포함되어 있다. 하드와이어드-파이프라인 회로는 데이터를 수신하고, 이 데이터를 메모리에 로드하고, 메모리로부터 데이터를 검색하고, 검색된 데이터를 처리하고, 그리고 처리된 데이터를 외부 소스로 제공하도록 동작 가능하다.

본 발명의 다른 실시예에 따르면, 하드와이어드-파이프라인 회로는 데이터를 수신하고, 수신된 데이터를 처리하고, 처리된 데이터를 메모리에 로드하고, 메모리로부터 상기 처리된 데이터를 검색하고, 그리고 상기 검색된 처리된 데이터를 외부 소스로 제공하도록 동작 가능하다.



파이프라인 가속기는 피어-벡터(peer-vector) 머신으로 일부로서 프로세서와 결합되어 있고, 메모리는 상기 하드웨어드-파이프라인 회로와 상기 프로세서가 처리하는 애플리케이션 간의 데이터 전송 - 단방향 또는 양방향 - 을 촉진한다.

#### 도면의 간단한 설명

도 1은 종래의 다중-프로세서 아키텍처를 갖는 컴퓨팅 머신의 블록 다이어그램이고,

도 2는 종래의 하드웨어드 파이프라인의 블록 다이어그램이고,

도 3은 본 발명의 일 실시예에 따른 피어-벡터 아키텍처를 갖는 컴퓨팅 머신의 블록 다이어그램이고,

도 4는 본 발명의 일 실시예에 따른 도 3의 파이프라인 가속기의 블록 다이어그램이고,

도 5는 본 발명의 일 실시예에 따른 도 4의 하드웨어드-파이프라인 회로 및 데이터 메모리의 블록 다이어그램이고,

도 6은 본 발명의 일 실시예에 따른 도 5의 통신 행의 메모리-기록 인터페이스의 블록 다이어그램이고,

도 7은 본 발명의 일 실시예에 따른 도 5의 통신 행의 메모리-관독 인터페이스의 블록 다이어그램이고,

도 8은 본 발명의 다른 실시예에 따른 도 3의 파이프라인 가속기의 블록 다이어그램이고,

도 9는 본 발명의 일 실시예에 따른 도 8의 하드웨어드-파이프라인 회로 및 데이터 메모리의 블록 다이어그램이다.

#### 실시예

##### (상세한 설명)

도 3은 본 발명의 일 실시예에 따른 피어-벡터 아키텍처를 갖는 컴퓨팅 머신(40)의 개략적인 블록 다이어그램이다. 호스트 프로세서(42)에 추가하여, 상기 피어-벡터 머신(40)에는 적어도 일부의 데이터를 수행하여 도 1의 컴퓨팅 머신(10) 내의 코프로세서(14)의 뱅크(bank)를 유효하게 대체하는 파이프라인 가속기(44)가 포함되어 있다. 따라서, 호스트-프로세서(42) 및 가속기(44)(또는 후술할 유닛)는 데이터 벡터를 앞뒤로 전송할 수 있는 "피어(peer)"이다. 가속기(44)는 프로그램 명령을 실행하지 않으므로, 가속기는 종종 코프로세서의 뱅크가 주어진 클럭 주파수에서 할 수 있는 것보다 훨씬 빠르게 데이터상의 집중적인 연산을 수학적으로 처리한다. 따라서, 프로세서(42)의 결정-형성 가능성과 가속기(44)의 수-크리닝 가능성을 결합함으로써, 머신(40)은 동일한 능력을 갖지만, 머신(10)과 같은 종래의 컴퓨팅 머신보다는 빠르게 데이터를 처리할 수 있다. 또한, 후술하는 바와 같이, 가속기(44)에 상기 호스트 프로세서(42)의 통신 인터페이스와 호환되는 통신 인터페이스를 제공하는 것은 머신(40)의 설계 및 수정을 용이하게 해 주고, 특히, 프로세서의 통신 인터페이스가 산업 표준인 경우 그러하다. 가속기(44)에 다수의 파이프라인 유닛(예를 들어, PLIC-기반 회로)이 포함되어 있는 경우, 이들 유닛 각각에 동일한 인터페이스를 제공하는 것은 가속기의 설계 및 수정을 용이하게 해 주는데, 특히 상기 통신 인터페이스가 산업-표준 인터페이스와 호환되는 경우 그러하다. 더욱이, 머신(40)은 후술하는 바와 같은 그리고 앞서 언급한 다른 출원들에서의 다른 장점도 제공한다.

도 3을 계속 참조하면, 호스트 프로세서(42) 및 파이프라인 가속기(44)에 추가하여, 피어-벡터 컴퓨팅 머신(40)에는 프로세서 메모리(46), 인터페이스 메모리(48), 버스(50), 펌웨어 메모리(52), 선택적인 원-데이터 입력 포트(54), 처리된-데이터 출력 포트(58), 및 선택적인 라우터(61)가 포함되어 있다.

호스트 프로세서(42)에는 처리 유닛(62) 및 메시지 처리기(64)가 포함되어 있으며, 프로세서 메모리(46)에는 처리-유닛 메모리(66) 및 처리기 메모리(68)를 포함하는데, 각각 프로세서 유닛 및 메시지 처리기를 위한 프로그램 및 작업 메모리 모두의 역할을 한다. 프로세서 메모리(46)에는 가속기-구성 레지스트리(70) 및 메시지-구성 레지스트리(72)도 포함되어 있는데, 이들은 각각 호스트 프로세서(42)로 하여금 가속기(44)의 기능 및 메시지 처리기(64)가 송수신하는 메시지의 포맷을 구성하도록 하는 각각의 구성 데이터를 저장하고 있다.

파이프라인 가속기(44)는 적어도 하나의 PLIC(도시하지 않음)에 배치되어 있으며 프로그램 명령을 실행하지 않고 각각의 데이터를 처리하는 하드와이어드 파이프라인(74<sub>1</sub>-74<sub>n</sub>)을 포함하고 있다. 펌웨어 메모리(52)는 가속기(44)용 구성 펌웨어를 저장한다. 만일 가속기(44)가 다수의 PLIC에 배치된다면, 이들 PLIC 및 그들 각각의 펌웨어 메모리는 다중 파이프라인 유닛(도 4)내에 배치된다. 상기 가속기(44)와 파이프라인 유닛은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR HAVING MULTIPLE PIPELINE UNITS AND RELATED COMPUTING MACHINE AND METHOD"인 미국 특허출원 제10/683,932호에 설명되어 있다. 선택적으로, 상기 가속기(44)는 적어도 하나의 ASIC에 배치될 수 있으며, 따라서, 구성할 수 없는 내부 상호접속을 갖는다. 이 대안에서, 머신(40)에서 펌웨어 메모리(52)를 생략할 수 있다. 또한, 비록 가속기(44)가 다중 파이프라인(74)을 포함하는 것으로 도시되어 있으나, 오직 하나의 파이프라인만 포함할 수 있다. 또한, 비록 도시하지는 않았지만, 가속기(44)에는 디지털-신호 처리기(DSP)와 같은 하나 또는 그 이상의 프로세서가 포함될 수 있다. 또한, 비록 도시하지는 않았지만, 가속기(44)에는 데이터 입력 포트 및/또는 데이터 출력 포트를 포함할 수 있다.

피어-백터 머신(40)의 일반적인 동작은 앞서 언급한 발명의 명칭이 "IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/684,102호에 설명되어 있고, 파이프라인 가속기(44)의 구조 및 동작은 도 4 내지 도 9를 참조하여 후술한다.

도 4는 본 발명의 일 실시예에 따른 도 3의 파이프라인 가속기(44)의 개략적인 블록 다이어그램이다.

가속기(44)에는 하나 또는 그 이상의 파이프라인 유닛(78)이 포함되어 있는데, 각각의 유닛에는 PLIC 또는 ASIC 같은 파이프라인 회로(80)가 포함되어 있다. 후술하고 상기 언급한 발명의 명칭이 "PIPELINE ACCELERATOR HAVING MULTIPLE PIPELINE UNITS AND RELATED COMPUTING MACHINE AND METHOD"인 미국 특허출원 제10/683,932호에 설명된 바와 같이, 각각의 파이프라인 유닛(78)은 호스트 프로세서(42) 및 가속기(44)의 다른 파이프라인 유닛의 "피어"이다. 즉, 파이프라인 유닛(78) 각각은 호스트 프로세서(42) 또는 다른 파이프라인 유닛과 직접 통신할 수 있다. 따라서, 이 피어-백터 아키텍처는 파이프라인 유닛(78) 모두가 마스터 파이프라인 유닛(도시하지 않음) 또는 호스트 프로세서(42)와 같은 중앙 위치를 통해서 통신하는 경우 발생하기도 하는 데이터의 "병목현상"을 막아준다. 또한, 설계자에게 머신을 크게 수정하지 않고 피어-백터 머신(40)(도 4)으로부터 피어를 추가 또는 제거할 수 있게 해준다.

파이프라인 회로(80)에는 호스트 프로세서(42)(도 3)와 같은 피어와 다음과 같은 파이프라인 회로의 다른 구성요소들 사이의 데이터를 전달해주는 통신 인터페이스(82)가 포함되어 있다. 상기 다른 구성요소들로는 통신 셀(84)을 통한 하드와이어드 파이프라인(74<sub>1</sub>-74<sub>n</sub>)(도 3), 제어기(86), 예외 관리자(88), 및 구성 관리자(80)가 있다. 파이프라인 회로(80)에는 산업-표준 버스 인터페이스(91)도 포함되어 있다. 선택적으로, 인터페이스(91)의 기능은 통신 인터페이스(82) 내에 포함되어도 좋다.

파이프라인 회로(80)의 구성요소들을 별개 모듈로 설계함으로써, 설계자는 파이프라인 회로의 설계를 간단하게 할 수 있다. 즉, 설계자는 이들 구성요소 각각을 개별적으로 설계 및 검사할 수 있으며, 소프트웨어 또는 프로세서-기반 컴퓨팅 시스템(도 1의 시스템(10)과 같은)을 설계하는 경우 이들을 하나로 집적할 수 있다. 또한, 설계자는 이들 구성요소, 특히 다른 파이프라인 설계에서 자주 사용될 통신 인터페이스(82)와 같은 구성요소를 정의하는 라이브러리 하드웨어 설명 언어(HDL)(도시하지 않음)를 세이브(save)할 수 있어서, 동일한 구성요소를 사용하는 미래의 파이프라인 설계를 검사하고 설계하는 시간을 줄여준다. 즉, 라이브러리에서 HDL을 사용함으로써, 설계자는 이전에 구현된 구성요소들을 "스크래치로부터" 다시 설계할 필요가 없어지므로, 이전에는 구현되지 못한 구성요소의 설계 또는 이전에 구현된 구성요소들의 수정에 노력을 집중할 수 있다. 더욱이, 파이프라인 회로(80) 또는 가속기(44)의 전체 파이프라인의 여러 버전(version)을 정의하는 라이브러리 HDL 내에 세이브할 수 있어서, 현존하는 디자인 가운데에서 고르고 선택할 수 있다.

통신 인터페이스(82)는 메시지 처리기(64)(도 3)에 의해 확인된 포맷으로 데이터를 송수신해서, 일반적으로 피어-백터 머신(40)(도 3)의 설계 및 수정을 용이하게 한다. 예를 들어, 만일 데이터 포맷이 Rapid I/O 포맷과 같은 산업 표준인 경우, 설계자는 호스트 프로세서(42)와 가속기(44) 사이의 맞춤형 접속을 설계할 필요는 없다. 또한, 논-버스 인터페이스를 통해서가 아니라 파이프라인 버스(50)를 통해서 호스트 프로세서(42)(도 3)와 같은 다른 피어와 파이프라인 회로(80)가 통신하도록 하게 함으로써, 설계자는 단지 이들을 파이프라인 유닛이 추가되거나 제거되는 시간마다 스크래치로부터 논-버스 인터페이스를 재설계하는 대신 파이프라인 버스와 이들(또는 이들을 보유하고 있는 회로 카드)을 접속하거나 접속해제하는 것만으로 파이프라인 유닛(78)의 수를 변경할 수 있다.

하드웨어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>)은 도 3을 참고하여 앞서 설명하고, 앞서 언급한 발명의 명칭이 "IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/684,102호에서 설명된 개별적으로 데이터상의 연산을 수행하며, 통신 셀(84)은 상기 파이프라인과 상기 파이프라인(80)의 다른 구성요소들 및 상기 파이프라인 회로의 외부에 있는 회로(후술하는 데이터 메모리(92)와 같은) 접속시킨다.

제어기(86)는 하드웨어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>)을 동기화 하고, 이들이 다른 피어들로부터의 통신, 즉 "이벤트"에 응답하여 각각의 데이터 연산을 수행하는 시퀀스를 모니터 및 제어한다. 예를 들어, 호스트 프로세서(42)와 같은 피어가 파이프라인 버스(50)를 통해 파이프라인 유닛(78)으로 이벤트를 송신하여 피어가 상기 파이프라인 유닛으로 데이터의 불릭 전송을 완료했음을 표시하고 하드웨어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>)이 이 데이터 처리를 시작하게 한다. 데이터를 포함하는 이벤트를 일반적으로 메시지로 부르며, 데이터를 포함하지 않는 이벤트를 "도어 벨(door bell)"이라고 부른다. 또한, 도 5를 참고로 후술하는 바와 같이, 파이프라인 유닛(78)은 동기화 신호에 응답하여 상기 파이프라인(74<sub>1</sub>~74<sub>n</sub>)을 동기화 하기도 한다.

예외 관리자(88)는 상기 하드웨어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>), 통신 인터페이스(82), 통신 셀(84), 제어기(86), 및 버스 인터페이스(91)의 상태를 모니터하고, 호스트 프로세서(42)(도 3)에게 예외를 보고한다. 예를 들어, 통신 인터페이스(82)내의 버퍼가 오버플로우(overflow) 되면, 예외 관리자(88)는 이를 호스트 프로세서(42)에게 보고한다. 예외 관리자는 상기 예외로 발생되는 문제를 해결하거나 또는 해결을 시도하기도 한다. 예를 들어, 버퍼의 오버플로우를 위해서, 예외 관리자(88)는, 후술하는 바와 같이, 구성 관리자(90)를 통해 또는 직접 버퍼의 크기를 증가시킨다.

구성 관리자(90)는, 앞서 언급한 발명의 명칭이 "IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/684,102호에 설명된 바와 같이, 호스트 프로세서(42)(도 3)로부터 소프트웨어-구성 데이터에 응답하여 하드웨어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>), 통신 인터페이스(82), 통신 셀(84), 제어기(86), 예외 관리자(88), 및 인터페이스(91)의 소프트웨어 구성(software configuration)을 설정하고, 하드 구성은 파이프라인 회로(80)의, 트랜지스터 및 회로-블럭 레벨상의 실제 토폴로지를 표시하고, 소프트웨어 구성은 상기 하드-구성된 구성요소의 물리적 파라미터(예를 들어, 데이터 폭, 데이터 크기)를 표시한다. 즉, 소프트웨어 구성 데이터는 프로세서의 연산 모드(예를 들어, 버스트-메모리 모드)를 설정하기 위해 프로세서의 레지스터(도 4에는 도시하지 않음)로 로드될 수 있는 데이터와 유사하다. 예를 들어, 호스트 프로세서(42)는 상기 구성 관리자(90)로 하여금 통신 인터페이스(82)내의 큐의 수 및 각각의 우선 레벨을 설정하도록 만드는 소프트웨어-구성 데이터를 송신한다. 예외 관리자(88)는 상기 구성 관리자(90)로 하여금, 예를 들어 통신 인터페이스(82) 내의 버퍼의 오버플로우 크기를 증가시키도록 하는 소프트웨어-구성 데이터를 송신하기도 한다.

도 4를 계속 참고하면, 파이프라인 회로(80)에 추가하여, 가속기(44)의 파이프라인 유닛(78)에는 데이터 메모리(92), 임의의 통신 버스(94), 및 상기 파이프라인 회로가 PLIC인 경우, 펌웨어 메모리(52)(도 3)를 포함한다.

데이터 메모리(92)는 데이터가 호스트 프로세서(42)(도 3)와 같은 다른 피어와 하드웨어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>) 사이에서 이동하도록 버퍼작용을 하고, 상기 하드웨어드 파이프라인을 위한 작업 메모리가 되기도 한다. 통신 인터페이스(82)는 데이터 메모리(92)와 파이프라인 버스(50)를 접속 시키고(통신 버스(94) 및 존재한다면 산업-표준 인터페이스(91)를 통해서), 통신 셀(84)은 데이터 메모리를 상기 하드웨어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>)과 접속시킨다.

산업-표준 인터페이스(91)는, 상기 통신 인터페이스로부터 인터페이스 회로의 일부를 유효하게 오프로드(offload)함으로써 상기 통신 인터페이스(82)의 크기 및 복잡성을 감소시키는 종래의 버스-인터페이스 회로이다. 그러므로, 설계자가 파이프라인 버스(50) 또는 라우터(61)(도 3)의 파라미터를 변경하고자 하는 경우에는, 통신 인터페이스(82)가 아니라 단지 인터페이스(91)를 수정하기만 하면 된다. 선택적으로, 설계자는 상기 인터페이스(91)를 상기 파이프라인 회로(80) 외부의 IC(도시하지 않음) 내에 배치해도 좋다. 상기 인터페이스(91)를 상기 파이프라인 회로(80)로부터 오프로드시키는 것은, 예를 들어, 하드웨어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>) 및 제어기(86)내에서의 사용을 위한 파이프라인 회로상의 리소스를 자유롭게 한다. 또는, 상기 설명한 바와 같이, 버스 인터페이스(91)는 통신 인터페이스(82)의 일부일 수도 있다.

도 3을 참고하여 위에서 설명한 바와 같이, 파이프라인 회로(80)는 PLIC 이고, 펌웨어 메모리(52)는 상기 파이프라인 회로의 하드 구성을 설정하는 상기 펌웨어를 저장한다. 메모리(52)는, 가속기(44)의 구성 동안에 상기 펌웨어를 파이프라인 회로(80)에 로드하고, 상기 가속기의 구성 동안에 또는 구성이 이루어진 후에 상기 통신 인터페이스(82)를 통해 호스트 프

로세서(42)(도 3)으로부터 수정된 펌웨어를 수신한다. 펌웨어를 로드하고 수신하는 것은 앞서 언급한 발명의 명칭이 "PROGRAMMABLE CIRCUIT AND RELATED COMPUTING MACHINE AND METHOD" 인 미국 특허출원 제10/684,057호에 더 설명되어 있다.

도 4를 계속 참고하면, 파이프라인 회로(80), 데이터 메모리(92) 및 펌웨어 메모리(52)는 더터 카드(daughter card)가 개인용 컴퓨터(도시하지 않음)내의 머더보드의 슬롯으로 꽂힐 수 있는 것치일 파이프라인-버스 커넥터(도시하지 않음)로 꽂히는 회로 보드 또는 카드(98)상에 배치된다. 비록 도시하지는 않았으나, 종래의 IC 및 전력 조정기(power regulator)나 전력 시퀀서와 같은 구성요소도 알려진 바와 같이 상기 카드(98)상에 배치되기도 한다.

상기 파이프라인 유닛(78)의 구조 및 동작의 보다 상세한 사항은 도 5를 참고하여 이하에 설명한다.

도 5는 본 발명의 일 실시예에 따른 도 4의 파이프라인 유닛(78)의 블록 다이어그램이다. 간략화를 위해서, 도 5에서는 펌웨어 메모리(52)를 생략했다. 파이프라인 회로(80)는 파이프라인 회로의 후술하는 구성요소들을 직접 또는 간접적으로 구동하는 마스터 CLOCK 신호를 수신한다. 파이프라인 회로(80)는 종래의 방법으로 상기 마스터 CLOCK 신호로부터 하나 또는 그 이상의 슬레이브 클럭 신호(도시하지 않음)를 발생한다. 파이프라인 회로(80)는 후술하는 바와 같이 동기화 신호(SYNC)를 수신하기도 한다.

데이터 메모리(92)에는 입력 듀얼-포트-상태-랜덤-액세스 메모리(DPSRAM)(100), 출력 DPSRAM(102), 및 선택적인 작업 메모리(104)가 포함되어 있다.

입력 DPSRAM(100)에는 통신 인터페이스(82)를 통해 호스트 프로세서(42)(도 3)와 같은 피어로부터 데이터를 수신하는 입력 포트(106)가 포함되어 있고, 통신 셀(84)을 통해 하드와이어드 파이프라인(74<sub>1</sub>-74<sub>n</sub>)으로 이 데이터를 제공하기 위한 출력 포트(108)가 포함되어 있다. 데이터 입력을 위한 하나와 데이터 출력을 위한 하나인 두 개의 포트를 가지는 것은, 상기 파이프라인(74<sub>1</sub>-74<sub>n</sub>)이 DPSRAM 으로부터 데이터를 판독하는 동안 상기 통신 인터페이스(82)가 데이터를 DPSRAM 으로부터 기록할 수 있기 때문에 상기 DPSRAM(100)으로/으로부터 데이터를 전송하는 속도 및 효율을 증가시킨다. 또한, 상기 설명한 바와 같이, 호스트 프로세서(42)와 같은 피어로부터 데이터를 버퍼하기 위해 DPSRAM(100)을 사용하는 것은 상기 피어 및 파이프라인(74<sub>1</sub>-74<sub>n</sub>)이 서로간에 상대적으로 동기되어 동작하도록 한다. 즉, 상기 피어는 현재 연산을 종료하기 위해 파이프라인을 위한 "대기" 없이 데이터를 상기 파이프라인(74<sub>1</sub>-74<sub>n</sub>)으로 전송할 수 있다. 유사하게, 파이프라인(74<sub>1</sub>-74<sub>n</sub>)은 데이터-전송 연산을 완료하기 위해 상기 피어를 위한 "대기" 없이 데이터를 검색할 수 있다.

유사하게, 출력 DPSRAM(102)에는 통신 셀(84)을 통해 상기 하드와이어드 파이프라인(74<sub>1</sub>-74<sub>n</sub>)으로부터 데이터를 수신하기 위한 입력 포트(110)가 포함되어 있고, 통신 인터페이스(82)를 통해 상기 호스트 프로세서(42)(도 3)와 같은 피어로 이 데이터를 제공하기 위한 출력 포트(112)가 포함되어 있다. 상기 설명한 바와 같이, 상기 두 개의 데이터 포트(110(입력) 및 112(출력))는 상기 DPSRAM(102) 로/로부터 데이터를 전송하는 속도 및 효율성을 증가시키고, 상기 파이프라인(74<sub>1</sub>-74<sub>n</sub>)으로부터 데이터를 버퍼하기 위해 DPSRAM(102)을 사용하는 것은 상기 피어 및 파이프라인이 서로간에 상대적으로 동기되어 동작하도록 한다. 즉, 상기 파이프라인(74<sub>1</sub>-74<sub>n</sub>)은 상기 피어로 또는 다른 피어로 데이터 전송을 완료하기 위해 상기 출력-데이터 처리기(126)를 위한 "대기" 없이 상기 피어로 데이터를 발행(publish)할 수 있다. 유사하게, 상기 출력-데이터 처리기(126)는 데이터-발행 연산을 완료하기 위해 상기 파이프라인(74<sub>1</sub>-74<sub>n</sub>)을 위한 "대기" 없이 피어로 데이터를 전송할 수 있다.

작업 DPSRAM(104)에는 통신 셀(84)을 통해 상기 하드와이어드 파이프라인(74<sub>1</sub>-74<sub>n</sub>)으로부터 데이터를 수신하기 위한 입력 포트(104)가 포함되어 있고, 통신 셀을 통해 상기 파이프라인으로 이 데이터를 퍼들러 보내기 위한 출력 포트(116)가 포함되어 있다. DPSRAM(100)으로부터 수신된 입력 데이터를 처리하는 동안, 상기 파이프라인(74<sub>1</sub>-74<sub>n</sub>)은 이 데이터의 처리를 계속하기 전에 부분적으로 처리된, 즉, 중간 데이터를 일시적으로 저장할 필요가 있을 수 있다. 예를 들어, 파이프라인(74<sub>1</sub>)과 같은 제1 파이프라인은 파이프라인(74<sub>2</sub>)과 같은 제2 파이프라인에 의한 추가 처리를 위한 중간 데이터를 발생한다. 따라서, 상기 제1 파이프라인은 상기 제2 파이프라인이 중간 데이터를 검색할 때 까지 상기 중간 데이터를 일시적으로 저장할 필요가 있다. 작업 SPDRAM(104)은 이러한 일시 저장소를 제공한다. 상기 설명한 바와 같이, 두 개의 데이터 포트(114(입력) 및 116(출력))는 파이프라인(74<sub>1</sub>-74<sub>n</sub>)과 DPSRAM(104) 사이의 데이터 전송의 속도 및 효율성을 증가시킨다. 또한, 개별적인 작업 DPSRAM(104)을 포함시키는 것은 상기 DPSRAM(100 및 102)가 데이터-입력 및 데이터-출

력 버퍼로서 각각 배타적으로 기능하도록 하게 함으로써 파이프라인 회로(80)의 속도 및 효율성을 증가시킨다. 그러나, 상기 파이프라인 회로(80)를 약간 수정하는 것은, 상기 DPSRAM(100) 및 DPSRAM(102) 중 어느 하나 또는 둘 모두가 DPSRAM(104)가 생략되는 경우 및 존재하는 경우, 상기 파이프라인(74<sub>1</sub>-74<sub>n</sub>)을 위한 작업 메모리일 수 있다.

비록 상기 DPSRAM(100,102,104)를 상기 파이프라인 회로(80)의 외부에 있는 것으로 설명하였으나, 이들 DPSRAM 중 하나 또는 그 이상, 또는 그와 동가가 상기 파이프라인 회로 내부에 있을 수도 있다.

도 5를 계속 참조하면, 통신 인터페이스(82)에는 산업-표준 버스 어댑터(118), 입력-데이터 처리기(120), 입력-데이터 및 입력-이벤트 큐(122,124), 출력-데이터 처리기(126), 및 출력-데이터 및 출력-이벤트 큐(128,130)가 포함되어 있다. 비록 상기 큐(122,124,128 및 130)가 하나의 큐로서 도시되어 있지만, 이들 큐의 하나 또는 그 이상에는 서브 큐(도시하지 않음)가 포함되기도 하는데, 상기 서브큐는 상기 큐 내에 저장된 값 또는 이들 값이 표현하는 데이터 각각을 우선순위에 의해 분리하게 한다.

산업-표준 버스 어댑터(118)에는 통신 버스(94)를 통해 상기 파이프라인 회로(80)와 상기 파이프라인 버스(50)(도 4) 사이에 데이터 전송을 하게 하는 물리 계층이 포함되어 있다. 따라서, 설계자가 버스(94)의 파라미터 변경을 원하는 경우, 설계자는 통신 인터페이스(82) 전체가 아니라 단지 어댑터(118)를 수정하기만 하면 된다. 산업-표준 버스 인터페이스(91)를 상기 파이프라인 유닛(78)에서 생략하면, 어댑터(118)를 수정하여 상기 파이프라인 버스(50)와 파이프라인 회로(80)간에 데이터 전송을 직접 하도록 한다. 후자를 수행하는 경우, 수정된 어댑터(118)에는 버스 인터페이스(91) 기능이 포함되며, 설계자는 버스(50)의 파라미터 변경을 원한다면 상기 어댑터(118)를 수정하기만 하면 된다.

입력-데이터 처리기(120)는 상기 산업-표준 어댑터(118)로부터 데이터를 수신하고, 이 데이터를 입력 포트(106)를 통해 DPSRAM(100)으로 로드하고, 상기 입력-데이터 큐(122)내의 데이터 및 대응하는 데이터 식별기로 포인터를 발생하여 저장한다. 단일 상기 데이터가 호스트 프로세서(42)(도 3)와 같은 피어로부터의 메시지의 페이로드(payload)라면, 상기 입력-데이터 처리기(120)는 상기 데이터를 DPSRAM(100)으로 로드하기 전에 상기 메시지에서 상기 데이터를 추출한다. 상기 입력-데이터 처리기(120)에는 상기 데이터를 DPSRAM(100)의 입력 포트(106)로 기록하는 인터페이스(132)가 포함되어 있는데, 이에 대해서는 도 6을 참고하여 아래에 설명한다. 선택적으로, 상기 입력-데이터 처리기(120)는 상기 추출 단계를 생략하고 DPSRAM(100)으로 메시지 전체를 로드할 수 있다.

상기 입력-데이터 처리기(120)는 또한 상기 산업-표준 버스 어댑터(118)로부터 이벤트를 수신하고, 이 이벤트를 상기 입력-이벤트 큐(124)로 로드한다.

또한, 상기 입력-데이터 처리기(120)에는 수신된 데이터 또는 이벤트가 파이프라인 회로(80)를 위한 것인지를 결정하는 검증 관리자(134)가 포함되어 있다. 이 검증 관리자(134)는 상기 데이터 또는 상기 이벤트를 포함하는 메시지의 헤더(또는 그 일부)를 분석함으로써, 데이터 또는 이벤트의 타입을 분석함으로써, 또는 상기 데이터 또는 이벤트의 순간 식별(즉, 상기 데이터/이벤트가 의도된 하드웨어드 파이프라인(74))의 분석으로 이 결정을 한다. 만약 상기 입력-데이터 처리기(120)은 상기 파이프라인 회로(80)를 위한 것이 아닌 데이터 또는 이벤트를 수신한다면, 상기 검증 관리자(134)는 상기 입력-데이터 처리기가 상기 수신된 데이터/이벤트를 로드하지 못하게 한다. 상기 피어-팩터 머신(40)에 상기 파이프라인 유닛(78)이 그 파이프라인 유닛을 위한 데이터/이벤트만을 수신해야 하도록 라우터(61)(도 3)가 포함되어 있으며, 상기 검증 관리자(134)는 상기 입력-데이터 처리기(120)로 하여금 상기 호스트 프로세서(42)(도 3)로 상기 예외(잘못 수신된 데이터/이벤트)를 식별하는 예외 메시지 및 상기 예외를 일으킨 피어를 송신하도록 한다.

상기 출력-데이터 처리기(126)는 상기 출력-데이터 큐(128)에 의해 지정된 상기 DPSRAM(102)의 위치로부터 처리된 데이터를 검색하고, 상기 처리된 데이터를 상기 산업-표준 버스 어댑터(118)를 통해 호스트 프로세서(42)(도 3)와 같은 하나 또는 그 이상의 피어로 전송한다. 상기 출력-데이터 처리기(126)에는 포트(112)를 통해 DPSRAM(102)로부터 상기 처리된 데이터를 판독하는 인터페이스(136)가 포함되어 있다. 상기 인터페이스(136)는 도 7을 참고하여 아래에 설명한다.

상기 출력-데이터 처리기(126)은 상기 파이프라인(74<sub>1</sub>-74<sub>n</sub>)에 의해 발생된 출력-이벤트 큐(130)로부터 이벤트를 검색하고 검색된 이벤트를 상기 산업-표준 버스 어댑터(118)를 통해 상기 호스트 프로세서(42)(도 3)와 같은 하나 또는 그 이상의 피어로 전송하기도 한다.

또한, 상기 출력-데이터 처리기(126)에는 상기 처리된 데이터 및 상기 이벤트를 서브스크라이브(subscribe)하는 상기 호스트 프로세서(42)(도 3)와 같은 피어 리스트를 포함하고 있는 서브스크립션 관리자(138)가 포함되어 있다. 상기 출력-데이터 처리기는 상기 리스트를 사용하여 상기 데이터/이벤트를 올바른 피어로 전송한다. 단일 피어가 상기 데이터/이벤트

가 메시지의 패킷로드에 있길 바란다면, 상기 출력-데이터 처리기(126)는 상기 서브스크립션 관리자(138)로부터 상기 피어의 네트워크 또는 버스-포트 어드레스를 검색하고, 상기 어드레스를 포함하는 헤더를 발생하고, 상기 데이터/이벤트 및 상기 헤더로부터 상기 메시지를 발생한다.

비록 상기 DPSRAM(100,102)내에 저장된 데이터를 저장하고 검색하는 기술이 포인터 및 데이터 식별기를 사용하는 것이 포함되어 있길 하지만, 설계자는 입력- 및 출력-데이터 처리기(120,126)를 수정하여 다른 데이터-관리 기술을 구현한다. 그러한 데이터-관리 기술의 종래 예에는 키나 토큰을 사용하는 포인터, 입력/출력 제어(IOC) 블록, 및 스텝링이 포함된다.

통신 셀(84)에는 상기 하드와이어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>)을 상기 출력-데이터 큐(128), 상기 제어기(86) 및 상기 DPSRAM(100,102,104)과 접속시키는 물리 계층이 포함된다. 상기 셀(84)에는 인터페이스(140,142) 및 선택적인 인터페이스(144,146)가 포함되어 있다. 상기 인터페이스(140,146)는 인터페이스(136)와 유사한데: 상기 인터페이스(140)는 포트(108)를 통해 DPSRAM(100)으로부터 입력 데이터를 판독하고, 상기 인터페이스(146)는 포트(116)를 통해 DPSRAM(104)로부터 중간 데이터를 판독한다. 상기 인터페이스(142,144)는 상기 인터페이스(132)와 유사한데: 상기 인터페이스(142)는 포트(110)를 통해 DPSRAM(102)으로 처리된 데이터를 기록하고, 상기 인터페이스(144)는 포트(114)를 통해 DPSRAM(104)으로 중간 데이터를 기록한다.

상기 제어기(86)에는 시퀀스 관리자(148) 및 동기화 인터페이스(150)가 포함되어 있는데, 상기 동기화 인터페이스는 하나 또는 그 이상의 동기화 신호(SYNK)를 수신한다. 호스트 프로세서(42)(도 3) 또는 피어-배터 머신(40)(도 3)의 외부에 있는 장치(기계하지 않음)와 같은 피어는, 후술하고 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR HAVING MULTIPLE PIPELINE UNITS AND RELATED COMPUTING MACHINE AND METHOD" 인 미국 특허출원 제10/683,932호의 하드와이어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>)을 활성화시키기 위해 상기 시퀀스 관리자를 트리거하는 SYNC 신호를 발생한다. 상기 동기화 인터페이스(150)는 SYNC 신호를 발생하여 상기 파이프라인 회로(80)를 트리거하거나 또는 다른 피어를 트리거한다. 또한, 상기 입력-이벤트 큐(124)로부터의 이벤트는 상기 시퀀스 관리자(148)를 트리거하여 후술하는 바와 같이 하드와이어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>)을 활성화시킨다.

시퀀스 관리자(148)는 통신 셀(84)을 통해 그들 각각의 연산을 통해 상기 하드와이어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>)을 순서화한다. 일반적으로, 파이프라인(74) 각각은 전처리, 처리 및 후처리의 적어도 세 개의 연산 상태를 가진다. 전처리가 진행되는 동안에는, 파이프라인(74)은 그 레지스터를 초기화하고 DPSRAM(100)으로부터 입력 데이터를 검색한다. 처리가 진행되는 동안에는, 파이프라인(74)은 상기 검색된 데이터를 처리하고 DPSRAM(104) 내에 중간 데이터를 임시로 저장하고, DPSRAM(104)에서 상기 중간 데이터를 검색하고, 그리고 상기 중간 데이터를 처리하여 결과 데이터를 발생한다. 후처리가 진행되는 동안에는, 파이프라인(74)은 상기 결과 데이터를 DPSRAM(102)으로 로드한다. 그러므로, 상기 시퀀스 관리자(148)는 파이프라인(74<sub>1</sub>~74<sub>n</sub>)의 동작을 모니터링하고 그의 동작 상태 각각이 시작되면 파이프라인 각각에게 명령한다. 설계자는 상기 설명한 것과는 다른 동작상태중에서 상기 파이프라인 태스크를 분배한다. 예를 들어, 파이프라인(74)은 상기 전처리 상태가 아니라 상기 처리 상태가 진행되는 동안 DPSRAM(100)으로부터 입력 데이터를 검색한다.

또한, 시퀀스 관리자(148)는 상기 하드와이어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>) 중에서 미리 결정된 내부 연산 동기화를 유지한다. 예를 들어, 파이프라인(74<sub>1</sub>~74<sub>n</sub>) 모두가 DPSRAM(100)으로부터 동시에 데이터 검색을 하는 것을 피하기 위해, 제1 파이프라인(74<sub>1</sub>)이 전처리 상태이고, 상기 제2 파이프라인(74<sub>2</sub>)이 처리 상태이고, 상기 제3 파이프라인(74<sub>3</sub>)이 후처리 상태인 동안 상기 파이프라인을 동기화 하도록 설계된다. 하나의 파이프라인(74)의 상태는 다른 파이프라인의 상태를 동시에 수행하는 것 보다는 클럭 사이클의 개수가 여러 개인 것을 요구하기 때문에, 파이프라인(74<sub>1</sub>~74<sub>n</sub>)은 단일 자유로운 구동이 허용된다면 동기화를 잃을 수도 있다. 그러므로, 어느 시간에서는, 예를 들어, 다수 파이프라인(74)이 DPSRAM(100)으로부터 데이터 검색을 동시에 시도하는 것과 같은 "병목현상"이 있을 수 있다. 동기화 손실 및 그것의 결과를 막기 위해서는, 상기 시퀀스 관리자(148)는 파이프라인(74) 모두로 하여금 어떠한 파이프라인이 다음 연산 상태로 가기 전에 현재의 연산 상태를 완료하게 한다. 따라서, 시퀀스 관리자(148)는 가장 느린 파이프라인(74)이 그 상태를 완료하도록 하기 위해 충분히 길도록 현재 연산 상태를 위한 할당을 한다. 선택적으로, 하드와이어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>) 중에서 미리결정된 연산 동기화를 유지하기 위한 회로(도시하지 않음)는 상기 파이프라인 자체 내부에 포함되기도 한다.

상기 하드웨어인드 파이프라인(74<sub>1</sub>~74<sub>n</sub>)을 시퀀스화하고 내부적으로 동기화하는 것에 대해서, 상기 시퀀스 관리자(148)는 상기 파이프라인의 동작을 호스트 프로세서(42)(도 3)와 같은 다른 피어의 연산, 및 하나 또는 그 이상의 SYNC 신호 또는 입력-이벤트 큐(124) 내의 이벤트에 응답하여 다른 외부 장치의 연산과 동기화 한다.

일반적으로, SYNC 신호는 시간-임계 기능을 트리거하지만 상당한 하드웨어 리소스를 필요로 하는데, 상대적으로, 이벤트는 비-시간-임계 기능을 트리거하지만 매우 적은 하드웨어 리소스를 요구한다. 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR HAVING MULTIPLE PIPELINE UNITS AND RELATED COMPUTING MACHINE AND METHOD"인 미국 특허출원 제10/682,932호에 설명된 바와 같이, SYNC 신호는 피어로부터 피어까지 직접 라우트되기 때문에, 파이프라인 버스(50)(도 3), 입력-데이터 처리기(120) 및 입력-이벤트 큐(124)를 통해 방식을 처리해야 하는 이벤트 보다 빠르게 기능을 트리거할 수 있다. 그러나, 이들은 개별적으로 라우트되기 때문에, SYNC 신호는 피어프라인 회로(80)의 라우팅 라인, 버퍼, 및 SYNC 인터페이스(150)와같은 전용 회로를 필요로 한다. 반대로, 이들은 현존하는 데이터-전송 기반시설(예를 들어, 파이프라인 버스(50) 및 입력-데이터 처리기(120))을 사용하기 때문에, 이벤트는 전용 입력-이벤트 큐(124)만을 요구한다. 따라서, 설계자는 최대의 시간-임계 기능만을 트리거하도록 이벤트를 사용하게 된다.

이하 설명은 트리거 기능의 예이다. 소나 센서 소자(도시하지 않음)가 파이프라인 유닛(78)으로 데이터의 물리를 전송한다고 가정하면, 입력-데이터 처리기(120)는 이 데이터를 DPSRAM(100)에 저장하고, 파이프라인(74<sub>1</sub>)은 이 데이터를 DPSRAM(100)로부터 DPSRAM(104)까지 전송하고, 트리거되면, 파이프라인(74<sub>2</sub>)은 DPSRAM(104)로부터 상기 데이터를 검색 및 처리한다. 만일 파이프라인(74<sub>2</sub>)이 상기 데이터상에서 수행하는 처리가 시간 임계라면, 상기 센서 소자는 SYNC 펄스를 발생하여 파이프라인(74<sub>1</sub>)이 데이터의 전체 물리를 DPSRAM(104)로 로드를 끝내는 즉시 인터페이스(150) 및 시퀀스 관리자(148)를 통해 파이프라인(74<sub>2</sub>)을 트리거한다. 파이프라인 유닛(78) 및 센서를 사용하여 파이프라인(74<sub>1</sub>)이 끝났는지를 결정하는 종래 기술은 많이 있다. 예를 들어, 후술하는 바와 같이, 시퀀스 관리자(148)는 대응하는 SYNC 펄스 또는 이벤트를 센서로 제공한다. 선택적으로, 만일 파이프라인(74<sub>2</sub>)이 수행하는 처리가 시간 임계가 아니라면, 센서는 파이프라인 버스(50)(도 3)를 통해 시퀀스 관리자(148)로 이벤트를 전송한다.

시퀀스 관리자(148)는 SYNC 펄스 또는 이벤트를 발생함으로써 하드웨어인드 파이프라인(74<sub>1</sub>~74<sub>n</sub>)의 연산에 관련된 정보를 호스트 프로세서(42)(도 3)와 같은 피어로 제공하기도 한다. 시퀀스 관리자(148)는 SYNC 인터페이스(150) 및 전용 선로(도시하지 않음)를 통해 SYNC 펄스를 전송하고, 출력-이벤트 큐(130) 및 출력-데이터 처리기(126)를 통해 이벤트를 전송한다. 상기 예를 참고하면, 피어가 파이프라인(74<sub>2</sub>)으로부터 상기 데이터 물리를 더 처리하는 것으로 가정한다, 시퀀스 관리자(148)는, 파이프라인(74<sub>2</sub>)이 데이터의 물리 처리를 끝내면 SYNC 펄스 또는 이벤트를 통해 상기 피어에게 통보한다. 시퀀스 관리자(148)는 또한 적절한 피어에게 대응하는 SYNC 펄스 또는 이벤트를 발생하고 전송함으로써 SYNC 펄스 또는 이벤트의 수신을 확인하기도 한다.

도 5를 계속 참고하면, 파이프라인 유닛(78)의 동작을 본 발명의 일 실시예에 따라 설명한다.

데이터를 위해, 산업-표준 버스 인터페이스(91)는 파이프라인 버스(50)(및 있다면 라우터(61))로부터 데이터 신호(호스트 프로세서(42)(도 3)와 같은 피어에서 생긴)를 수신하고, 이들 신호를 헤더 및 페이로드를 각각 가지는 메시지로 변환(translate)한다.

다음으로, 산업-표준 버스 어댑터(118)가 이 메시지를 상기 산업-표준 버스 인터페이스(91)를 통해 입력-데이터 처리기(120)와 호환되는 포맷으로 바꾼다.

그 다음, 입력-데이터 처리기(120)는 메시지 헤더를 조사하여 각 헤더로부터 데이터 페이로드를 설명하는 부분을 추출한다. 예를 들어, 추출된 헤더 부분에 파이프라인 유닛(78)의 어드레스, 페이로드내의 데이터의 타입, 또는 데이터가 의도된 파이프라인(74<sub>1</sub>~74<sub>n</sub>)을 식별하는 순간 식별자 등이 포함될 수 있다.

다음으로, 검증 관리자(134)가 상기 추출된 헤더 부분을 분석하고 상기 하드웨어인드 파이프라인(74<sub>1</sub>~74<sub>n</sub>)의 하나로 의도된 데이터를 확인하며, 상기 인터페이스(132)는 이 데이터를 포트(106)를 통해 DPSRAM(100)의 위치로 기록하며, 상기 입력-데이터 처리기(120)가 상기 입력-데이터 큐(122) 내의 대응하는 데이터 식별기 및 상기 위치로 포인터를 저장한다. 상기 데이터 식별기는 데이터가 의도된 파이프라인 또는 파이프라인들(74<sub>1</sub>~74<sub>n</sub>)을 식별하거나 또는 시퀀스 관리자

(148)로 하여금 이 식별을 추출하는 바와 같이 만드는 정보를 포함한다. 선택적으로, 상기 큐(122)에는 파이프라인(74<sub>1</sub>~74<sub>n</sub>) 각각을 위한 개별적인 서브큐를 포함하기도 하고, 상기 입력-데이터 처리기(120)는 상기 의도된 파이프라인 또는 파이프라인들의 서브큐 또는 서브큐들 내의 포인터를 저장한다. 이 대안에서, 상기 데이터 식별기는 생략해도 좋다. 또한, 상기 데이터가 메시지의 페이로드인 경우에는, 입력-데이터 처리기(120)가 상기 인터페이스(132)가 DPSRAM(100)내의 데이터를 저장하기 전에 상기 메시지에서부터 데이터를 추출한다. 선택적으로, 상기 설명한 바와 같이, 인터페이스(132)는 DPSRAM(100)내의 전체 메시지를 저장한다.

다음으로, 적절한 시간에서, 시퀀스 관리자(148)는 상기 입력-데이터 큐(122)로부터 상기 포인터와 상기 데이터 식별기를 판독하고, 상기 데이터 식별기로부터 상기 데이터가 의도된 파이프라인 또는 파이프라인들(74<sub>1</sub>~74<sub>n</sub>)을 결정하고, 상기 포인터를 통신 셀(84)을 통해 상기 파이프라인 또는 파이프라인들(74<sub>1</sub>~74<sub>n</sub>)로 패스시킨다.

다음으로, 상기 데이터-수신 파이프라인 또는 파이프라인들(74<sub>1</sub>~74<sub>n</sub>)은 상기 인터페이스(140)으로 하여금 DPSRAM(100)의 지정된 위치에서 상기 데이터를 검색하게 한다.

이어서, 상기 데이터-수신 파이프라인 또는 파이프라인들(74<sub>1</sub>~74<sub>n</sub>)은 상기 검색된 데이터를 처리하고, 인터페이스(142)는 그 처리된 데이터를 포트(110)를 통해 DPSRAM(102)의 위치에 기록하고, 통신셀(84)은 상기 출력-데이터 큐(128)로 포인터 및 상기 처리된 데이터를 위한 데이터 식별기를 로드한다. 상기 데이터 식별기는 상기 처리된 데이터를 서브스크립트하는 호스트 프로세서(42)(도 3)와 같은 목적지 피어 또는 피어들을 식별하거나 또는 서브스크립션 관리자(138)가 상기 목적지 피어 또는 피어들(예를 들어, 호스트 프로세서(42)(도 3))을 순차적으로 결정하게 하는 정보(데이터 타입 등)를 포함한다. 선택적으로, 상기 큐(128)에는 각각의 파이프라인(74<sub>1</sub>~74<sub>n</sub>)을 위한 개별적인 서브큐(도시하지 않음)가 포함되기도 하며, 상기 통신셀(84)이 원래의 파이프라인 또는 파이프라인들의 서브큐 또는 서브큐들 내의 포인터를 저장한다. 이 대안에서, 통신셀(84)은 큐(128)로 데이터 식별기를 로드하는 것을 생략해도 좋다. 또한, 만일 상기 파이프라인 또는 파이프라인들(74<sub>1</sub>~74<sub>n</sub>)이 검색된 데이터가 처리되는 동안 중간 데이터를 발생한다면, 인터페이스(144)가 포트(114)를 통해 이 중간 데이터를 DPSRAM(104)에 기록하고, 인터페이스(146)가 포트(116)를 통해 DPSRAM(104)로부터 이 중간 데이터를 검색한다.

다음으로, 출력-데이터 처리기(126)은 상기 출력-데이터 큐(128)로부터 포인터 및 데이터 식별기를 검색하고, 서브스크립션 관리자(138)가 상기 식별기로부터 상기 데이터의 목적지 피어 또는 피어들(예를 들어, 도 3의 호스트 프로세서(42))을 결정하고, 인터페이스(136)는 포트(112)를 통해 DPSRAM(102)의 지정된 위치로부터 상기 데이터를 검색하고, 출력-데이터 처리기는 이 데이터를 산업-표준 버스 어댑터(118)로 전송한다. 만일 목적지 피어가 메시지의 페이로드일 데이터를 필요로 한다면, 출력-데이터 처리기(126)는 메시지를 발생하고 이 메시지를 어댑터(118)로 전송한다. 예를 들어, 데이터에 다수의 목적지 피어가 있다고 가정하면, 파이프라인 버스(50)는 메시지 브로드캐스팅(broadcasting)을 지원한다. 상기 출력-데이터 처리기(126)는 모든 목적지 피어의 어드레스를 포함하는 하나의 헤더를 발생하고, 이 헤더와 데이터를 메시지와 결합하고, 모든 목적지 피어에게 하나의 메시지를 동시에 전송(어댑터(118) 및 산업-표준 버스 인터페이스(91)를 통해)한다. 선택적으로, 출력-데이터 처리기(126)가 각각의 헤더를 발생하고, 그러므로써 각각의 목적지 피어를 위한 개별적인 메시지를 발생하고 이를 각각 전송한다.

이어서, 산업-표준 버스 어댑터(118)는 출력-데이터 처리기(126)로부터 데이터를 포맷하여 산업-표준 버스 인터페이스(91)에 호환되도록 한다.

다음으로, 상기 산업-표준 버스 인터페이스(91)는 산업-표준 버스 어댑터(118)를 통해 상기 데이터를 포맷하여 파이프라인 버스(50)와 호환되도록 한다(도 3).

도어벨 같은 일부 데이터가 없는 이벤트를 위해서는, 산업-표준 버스 인터페이스(91)는 파이프라인 버스(50)(및 존재한다면 라우터(61))로부터 신호(도 3의 호스트 프로세서와 같은 피어로부터 발생됨)를 수신하고, 이 신호를 상기 이벤트를 포함하는 헤더로 변환(즉, 데이터없는 메시지)한다.

다음으로, 산업-표준 버스 어댑터(118)는 상기 산업-표준 버스 인터페이스(91)를 통해 상기 헤더를 입력-데이터 처리기(120)와 호환되는 포맷으로 바꾼다.



이어서, 상기 입력-데이터 처리기(120)는 상기 헤더로부터 상기 이벤트 및 상기 이벤트의 디스크립션을 추출한다. 예를 들어, 상기 디스크립션에는 파이프라인 유닛(78)의 어드레스, 이벤트 타입, 또는 상기 이벤트가 의도된 파이프라인(74<sub>1</sub>~74<sub>n</sub>)을 식별하는 순간 식별기가 포함될 수 있다.

다음으로, 검증 관리자(134)가 상기 이벤트 디스크립션을 분석하고 상기 이벤트가 하드와이어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>)의 하나로 의도된 것임을 확인하고, 상기 입력-데이터 처리기(120)는 상기 이벤트 및 상기 입력-이벤트 큐(124) 내의 그것의 디스크립션을 저장한다.

이어서, 적절한 시간에서, 시퀀스 관리자(148)가 상기 입력-이벤트 큐(124)로부터 상기 이벤트 및 그것의 디스크립션을 관독하고, 그 이벤트에 응답하여, 상기 설명된 바와 같이 파이프라인(74<sub>1</sub>~74<sub>n</sub>)의 하나 또는 그 이상의 연산을 트리거한다. 예를 들어, 시퀀스 관리자(148)는 파이프라인(74<sub>2</sub>)을 트리거하여 DPSRAM(104) 내에 앞서 저장된 파이프라인(74<sub>1</sub>)의 데이터 처리를 시작한다.

이벤트를 출력하기 위하여, 시퀀스 관리자(148)는 이벤트와 이벤트의 디스크립션을 발생시키고 출력 이벤트 큐(130)에 이벤트와 그 디스크립션을 로드한다 - 하나 이상의 가능한 목적지 피어가 있는 경우 이벤트 디스크립션을 이벤트용 목적지 피어를 확인한다. 예를 들어, 위에서 설명된 바와 같이, 이벤트는 입력 이벤트, 입력 데이터 또는 입력 이벤트 메시지, 또는 SYNC 펄스의 수신과 수행을 확인할 수 있다.

다음으로, 출력 데이터 처리기(126)는 이벤트와 출력 이벤트 큐(130)로부터의 그 디스크립션을 회수하고, 서브스크립션 관리자(138)는 이벤트 디스크립션으로부터 이벤트의 목적지 피어 또는 피어들(예를 들어, 도3의 호스트 프로세서(42))을 결정하며, 출력 데이터 처리기는 이벤트를 상기에서 설명한 산업 표준 버스 어댑터(118)와 산업 표준 버스 인터페이스(91)를 경유하여 적절한 목적지 피어 또는 피어들에 전송한다.

구성 명령으로서, 산업 표준 버스 어댑터(118)는 산업 표준 버스 인터페이스(91)를 경유하여 호스트 프로세서(42)(도3)로부터 명령을 수신하고, 데이터가 없는 이벤트(즉, 도어벨)용으로 상기의 설명과 유사한 방식으로 입력 데이터 처리기(120)에 명령을 제공한다.

다음으로, 검증 관리자(134)는 명령이 파이프라인 유닛(78)용이라는 것과 입력 데이터 처리기(120)이 구성 관리자(90)에 명령을 로드하는 것을 확인한다. 더욱이, 입력 데이터 처리기(120) 또는 구성 관리자(90)는 명령을 출력 데이터 처리기(126)에 또한 통과시켜, 파이프라인 유닛(78)이 명령을 전송한 피어(예를 들어, 도3의 호스트 프로세서(42))에 명령을 되돌려 보내어 명령을 받았다는 것을 확인한다. 이러한 확인 기술을 종종 “에코잉(echoing)”이라고 불려진다.

다음으로, 구성 관리자(90)는 명령을 수행한다. 예를 들어, 명령은 디버깅 목적으로 파이프라인(74<sub>1</sub>~74<sub>n</sub>)중의 하나를 디스이블(disable)시킬 수 있다. 또는, 명령은 호스트 프로세서(42)(도3)와 같은 피어를 허용하여 출력 데이터 처리기(126)를 경유하여 구성 관리자(90)로부터 파이프라인 회로(80)의 현재 구성을 읽게 할 수 있다. 게다가, 예외 관리자(88)에 의하여 인식된 예외를 정리하게 구성 명령을 사용할 수 있다.

예외로서, 파이프라인 회로(80)의 입력 데이터 큐(122)와 같은 구성요소는 예외 관리자(88)에 예외를 트리거한다. 하나의 수행으로, 구성요소는 그 구성요소를 모니터링하는 예외 트리거링 어댑터(도시하지 않음)를 포함하고 소정의 조건 또는 조건 세트에 응하여 예외를 트리거한다. 예외 트리거링 어댑터는 하나로 설계되고 예외를 발생시킬 수 있는 파이프라인 회로(80)의 각 구성요소의 부품을 포함할 수 있는 유니버설 회로일 수 있다.

다음으로, 예외 트리거에 응답하여, 예외 관리자(88)는 예외 식별기를 발생시킨다. 예를 들어, 식별기는 입력 데이터 큐(122)가 오버플로우 되는 것을 나타낼 수 있다. 더욱이, 식별기는 하나 이상의 가능 목적지 피어가 있는 경우, 목적지 피어를 포함할 수 있다.

그리고 나서, 출력 데이터 처리기(126)는 예외 관리자(88)로부터 예외 식별기를 회수하고, 앞서 인용한 발명의 명칭이 “COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD”인 미국 특허출원 제10/684,053호에 설명된 바와 같이 예외 식별기를 호스트 프로세서(42)(도3)에 전송한다.

다른 예로서, 다중 가능 목적지 피어가 있는 경우, 예의 식별기는 목적지 정보도 포함하여 서브스크립션 관리자(138)가 식별기의 목적지 피어 또는 피어들(예를 들어, 도3의 호스트 프로세서(42))을 결정한다. 출력 데이터 처리기(126)는 식별기를 산업-표준 버스 어댑터(118)와 산업-표준 버스 인터페이스(91)를 경유하여 목적지 피어 또는 피어들에게 전송한다.

도 5를 참조하면, 파이프라인 유닛(78)의 또 다른 실시예가 나타나 있다. 예를 들어, DPSRAM 을 포함하는 것으로 도시되어 있으나, 데이터 메모리(92)는 퀴드 데이터 레이트(QDR) SRAM 과 같은 다른 형태의 메모리 IC를 포함할 수 있다.

도 6은 본 발명의 일실시예에 따른 도 5의 인터페이스(142)의 블록도이다. 도 5와 함께 설명된 바와 같이, 인터페이스(142)는 하드와이어드 파이프라인(74<sub>1</sub>-74<sub>n</sub>)에서 DPSRAM(102)로 처리된 데이터를 기록한다. 아래에서 설명하는 바와 같이, 인터페이스(142)의 구조는 데이터 "병목" 을 감소시키거나 줄이고, 파이프라인 회로(80)(도 5)가 PLIC인 경우, PLIC의 로컬 및 글로벌 라우팅 리소스를 효율적으로 사용하게 한다.

인터페이스(142)는 기록 채널(150<sub>1</sub>-150<sub>n</sub>), 각각의 하드와이어드 파이프라인(74<sub>1</sub>-74<sub>n</sub>)(도 5), 및 제어기(152)를 포함한다. 예시적으로, 채널(150)만이 아래에서 설명되고, 다른 채널(150<sub>2</sub>-150<sub>n</sub>)의 동작과 구조는 특별히 언급하지 않는 한 유사한 것으로 이해하면 된다.

채널(150<sub>1</sub>)은 기록-어드레스/데이터 FIFO(154<sub>1</sub>)와 어드레스/데이터 레지스터(156<sub>1</sub>)를 포함한다.

FIFO(154<sub>1</sub>)는 파이프라인(74<sub>1</sub>)이 DPSRAM(102)에 쓴 데이터를 저장하고, 제어기(152)가 실제로 레지스터(156<sub>1</sub>)을 경유하여 DPSRAM(102)에 데이터를 기록할 수 있을 때까지 파이프라인이 데이터를 기록한 DPSRAM(102)내에 위치 어드레스를 저장한다. 그리하여, FIFO(154<sub>1</sub>)는 제어기(152)가 앞의 데이터 기록을 완료할 때까지 파이프라인(74<sub>1</sub>)이 채널(150<sub>1</sub>)에 데이터를 쓰기위하여 대기해야하는 경우 일어나는 데이터 병목을 줄이거나 제거한다.

FIFO(154<sub>1</sub>)는 버스(158)를 경유하여 파이프라인(74<sub>1</sub>)으로부터 데이터를 수신하고, 데이터가 버스(160<sub>1</sub>)를 경유하여 기록될 위치 어드레스를 수신하며, 데이터와 어드레스를 버스(162<sub>1</sub>과 164<sub>1</sub>)를 각각 경유하여 레지스터(156<sub>1</sub>)에 데이터와 어드레스를 제공한다. 더욱이, FIFO(154<sub>1</sub>)는 라인(166<sub>1</sub>)상에서 파이프라인(74<sub>1</sub>)으로부터 WRITE FIFO 신호를 받고, 라인(168<sub>1</sub>)을 경유하여 CLOCK 신호를 받으며, FIFO FULL 신호를 라인(170<sub>1</sub>)상에서 파이프라인(74<sub>1</sub>)에 제공한다. 게다가, FIFO(154<sub>1</sub>)는 라인(172<sub>1</sub>)을 경유하여 제어기(152)로부터 READ FIFO 신호를 받고, 라인(174<sub>1</sub>)을 경유하여 제어기에 FIFO EMPTY 신호를 제공한다. 파이프라인 회로(80)(도 5)가 PLIC인 경우, 버스(158<sub>1</sub>, 160<sub>1</sub>, 162<sub>1</sub>, 및 164<sub>1</sub>)와 라인(166<sub>1</sub>, 168<sub>1</sub>, 170<sub>1</sub>, 172<sub>1</sub>, 및 174<sub>1</sub>)은 로컬 라우팅 리소스를 활용하여 바람직하게 형성된다. 전형적으로, 신호 경로 길이는 대체로 더 짧고 라우팅이 수행하기 더 쉽기 때문에 글로벌 라우팅 리소스보다 바람직하다.

레지스터(156<sub>1</sub>)은 버스(162<sub>1</sub>, 164<sub>1</sub>)를 각각 경유하여 FIFO(154<sub>1</sub>)로부터 쓰기 위치의 어드레스와 기록될 데이터를 수신하고 데이터와 어드레스를 어드레스/데이터 버스(176)를 경유하여 DPSRAM(102)(도 5)의 포트(110)에 제공한다. 더욱이, 레지스터(156<sub>1</sub>)는 아래에서 설명되는 바와 같이 어드레스/데이터 버스(178)를 경유하여 레지스터(156<sub>2</sub>-156<sub>n</sub>)로부터 데이터와 어드레스를 또한 받는다. 게다가, 레지스터(156<sub>1</sub>)는 라인(180)을 경유하여 제어기(152)로부터 SHIFT/LOAD 신호를 받는다. 파이프라인 회로(80)(도 5)가 PLIC인 경우에 버스(170)는 글로벌 라우팅 리소스를 사용하여 전형적으로 형성되고, 버스(178<sub>1</sub>-178<sub>n-1</sub>)와 라인(180)은 로컬 라우팅 리소스를 활용하여 바람직하게 형성된다.

FIFO EMPTY 신호를 받고 READ FIFO와 SHIFT/LOAD 신호를 발생시킴과 더불어 제어기(152)는 WRITE DPSRAM 신호를 라인(182)을 경유하여 DPSRAM(102)(도 5)의 포트(110)에 제공한다.

도 6을 참조하여, 인터페이스(142)의 동작을 살펴보면 다음과 같다.

우선, FIFO(154<sub>1</sub>)는 FIFO의 현재 상태("풀(full)" 또는 "풀 아님(not full)")에 대응하여 로직 레벨로 FIFO FULL 신호를 구동한다.

다음으로, FIFO(154<sub>1</sub>)가 풀 아님이고 파이프라인(74<sub>1</sub>)가 기록될 데이터를 처리하는 경우, 파이프라인은 버스(158<sub>1</sub>, 160<sub>1</sub>) 각각에 데이터와 대응 어드레스를 구동시키고, WRITE 신호를 행사하여, FIFO에 데이터와 어드레스를 로딩한다. 그러나, FIFO(154<sub>1</sub>)가 풀인 경우, 파이프라인(74<sub>1</sub>)은 FIFO가 데이터를 로딩하기 전에 풀 아님일 때까지 대기한다.

그 다음, FIFO(154<sub>1</sub>)는 FIFO의 현재 상태("엠프티(empty)" 또는 "엠프티 아님(not empty)")에 대응하여 로직 레벨로 FIFO EMPTY 신호를 구동시킨다.

다음으로, FIFO(154<sub>1</sub>)가 엠프티 아님인 경우, 제어기(152)는 READ FIFO 신호를 행사하고 SHIFT/LOAD 신호를 로직 레벨로 구동시킨다. FIFO에서 레지스터(156<sub>1</sub>)로 첫 번째 로드된 데이터와 어드레스를 로딩한다. FIFO(154<sub>1</sub>)가 엠프티이면, 제어기(152)는 READ FIFO를 행사하지 않고, 다른 FIFO(152<sub>2</sub>~154<sub>n</sub>)의 어느 것이 엠프티 아님인 경우 로직 레벨로 SHIFT 로드로 구동시킨다.

채널(150<sub>2</sub>~150<sub>n</sub>)은 FIFO(154<sub>1</sub>~154<sub>n</sub>)에 첫 번째 로드된 데이터가 레지스터(156<sub>2</sub>~156<sub>n</sub>)에 각각 로드되는 것과 유사한 방식으로 동작한다.

그 다음, 제어기(152)는 시프트 로직 레벨로 SHIFT/LOAD 신호를 구동하고 WRITE DPSRAM 신호를 행사하여, 레지스터(156<sub>1</sub>~156<sub>n</sub>)로부터 어드레스/데이터 버스(176)상에 데이터와 어드레스를 순차적으로 시프트시키고 DPSRAM(102)의 대응 위치에 데이터를 로드시킨다. 특히, 첫 번째 시프트 사이클 동안에, 레지스터(156<sub>1</sub>)로부터 데이터와 어드레스는 버스(176)상에 시프트되어 FIFO(154<sub>2</sub>)로부터 데이터는 DPSRAM(102)의 어드레스화된 위치에 로드된다. 또한 첫 번째 사이클 동안에, 레지스터(156<sub>2</sub>)로부터 데이터와 어드레스는 레지스터(156<sub>1</sub>)로 시프트되고, 레지스터(156<sub>2</sub>)(도시하지 않음)로부터 데이터와 어드레스는 레지스터(156<sub>2</sub>)로 시프트되고 나머지도 동일하다. 두 번째 시프트 사이클 동안에 레지스터(156<sub>1</sub>)로부터 데이터와 어드레스는 버스에 시프트되어 FIFO(154<sub>2</sub>)로부터 데이터가 DPSRAM(102)의 어드레스화된 위치에 로드된다. 또한, 두 번째 시프트 사이클 동안에 레지스터(156<sub>1</sub>)로부터 데이터와 어드레스는 레지스터(156<sub>1</sub>)에 시프트되고, 레지스터(156<sub>2</sub>)(도시하지 않음)로부터 데이터와 어드레스는 레지스터(156<sub>1</sub>)로 시프트되고 나머지도 같다. n 시프트 사이클이 있고 n 번째 시프트 사이클 동안에 레지스터(156<sub>n</sub>)로부터의 데이터와 어드레스(FIFO(154<sub>n</sub>)로부터의 데이터와 어드레스임)은 버스(176)상으로 시프트된다. 제어기(152)는 SHIFT/LOAD 신호를 펄스하거나 레지스터(156<sub>1</sub>~156<sub>n</sub>)에 결합된 시프트 클럭 신호(도시하지 않음)를 발생하여 시프트 사이클을 수행할 수 있다. 더욱이, 제어기(152)가 레지스터를 로드된 경우 대응 FIFO(154<sub>1</sub>~154<sub>n</sub>)가 엠프티이기 때문에 레지스터(156<sub>1</sub>~156<sub>n</sub>)중 하나가 특정 시프트 동작 동안에 엠프티인 경우, 제어기는 엠프티 레지스터를 바이패스할 수 있어 버스(176)상에 영 데이터와 영 어드레스를 시프트하는 것을 피하여 시프트 동작을 단축시킨다.

도5와 도6을 참조하여, 본 발명의 일 실시예에 따르면, 인터페이스(144)는 인터페이스(142)와 유사하고, 인터페이스(132)는 하나의 기록 채널만을 포함하는 것을 제외하고는 인터페이스(142)와 또한 유사하다.

도7은 본 발명의 일 실시예에 따른 도5의 인터페이스(140)의 블록도이다. 도5와 함께 위에서 설명된 바와 같이, 인터페이스(140)는 DPSRAM(100)으로부터 입력 데이터를 읽고 이 데이터를 하드와이어드(74<sub>1</sub>~74<sub>n</sub>)에 전송한다. 아래에서 설명되는 바와 같이, 인터페이스(140)의 구조는 데이터 "병목"을 줄이거나 제거하고, 파이프라인 회로(80)(도5)가 PLIC인 경우 PLIC의 로컬과 글로벌 라우팅 리소스를 효율적으로 활용한다.

인터페이스(140)는 판독 채널(190<sub>1</sub>~190<sub>n</sub>), 각 하드와이어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>)을 하나의 채널(도 5) 및 제어기(192)를 포함한다. 예시적으로, 채널(190<sub>1</sub>)만이 아래에서 설명되고, 다른 채널(190<sub>2</sub>~190<sub>n</sub>)의 동작과 구조는 특별히 언급하지 않는 한 유사한 것으로 이해하면 된다.

채널(190<sub>1</sub>)은 FIFO(194<sub>1</sub>)와 어드레스/식별기(ID) 레지스터(196<sub>1</sub>)를 포함한다. 아래에서 설명하는 바와 같이, 식별기는 파이프라인(74<sub>1</sub>~74<sub>n</sub>)을 식별하여 데이터를 받기위하여 DPSRAM(100)의 특정 위치에서 데이터를 판독하도록 요청한다.

FIFO(194<sub>1</sub>)는 두개의 서브 FIFO(도시하지 않음)를 포함하여, 하나는 파이프라인(74<sub>1</sub>)이 입력 데이터를 읽고자하는 DPSRAM(100)의 위치 어드레스를 저장하고, 다른 하나는 DPSRAM(100)으로부터 읽은 데이터를 저장한다. 이리하여, 제어기(192)가 앞의 데이터 판독을 완료할 때까지 파이프라인(74<sub>1</sub>)이 채널(190<sub>1</sub>)에 판독한 어드레스를 제공하기 위하여 제어기(192)가 대기해야하는 경우나 제어기가 후속 데이터를 판독하기 전에 파이프라인(74<sub>1</sub>)이 판독한 데이터를 회수하기 까지 제어기가 대기해야하는 경우 FIFO(194<sub>1</sub>)는 일어날 수 있는 병목을 줄이거나 제거한다.

FIFO(194<sub>1</sub>)는 버스(198<sub>1</sub>)를 경유하여 파이프라인(74<sub>1</sub>)으로부터 판독한 어드레스를 수신하고 버스(200<sub>1</sub>)를 경유하여 레지스터(196<sub>1</sub>)에 어드레스와 ID를 제공한다. ID는 파이프라인(74<sub>1</sub>)에 대응하고 대체로 변경되지 않기 때문에, FIFO(194<sub>1</sub>)는 ID를 저장하고 ID와 어드레스를 연결시킨다. 다른 예로서, 파이프라인(74<sub>1</sub>)은 버스(198<sub>1</sub>)를 경유하여 FIFO(194<sub>1</sub>)에 ID를 제공할 수 있다. 더욱이, FIFO(194<sub>1</sub>)는 라인(202<sub>1</sub>)을 경유하여 파이프라인(74<sub>1</sub>)으로부터 READY WRITE FIFO 신호를 받고, 라인(204<sub>1</sub>)을 경유하여 제어기에 CLOCK 신호를 제공하며, (읽은 어드레스의) FIFO FULL 신호를 라인(206<sub>1</sub>)을 경유하여 파이프라인에 제공한다. 게다가, FIFO(194<sub>1</sub>)은 라인(208<sub>1</sub>)을 경유하여 제어기(192)로부터 WRITE/READ 신호를 제공하고 라인(210<sub>1</sub>)을 경유하여 제어기에 FIFO EMPTY 신호를 제공한다. 더욱이, FIFO(194<sub>1</sub>)는 버스(212)를 경유하여 제어기(192)로부터 대응 ID와 읽은 데이터를 받고, 버스(214<sub>1</sub>)를 경유하여 파이프라인(74<sub>1</sub>)에 이 데이터를 제공한다. 파이프라인 회로(80)(도 5)가 PLIC인 경우, 버스(198<sub>1</sub>, 200<sub>1</sub>, 및 214<sub>1</sub>)와 라인(202<sub>1</sub>, 204<sub>1</sub>, 206<sub>1</sub>, 208<sub>1</sub>, 및 210<sub>1</sub>)은 로컬 라우팅 리소스를 활용하여 바람직하게 형성되고, 버스(212)는 글로벌 라우팅 리소스를 활용하여 전형적으로 형성된다.

레지스터(196<sub>1</sub>)는 버스(206<sub>1</sub>)를 경유하여 FIFO(194<sub>1</sub>)로부터 대응 ID와 판독 위치의 어드레스를 수신하고, 어드레스 버스(216)를 경유하여 DPSRAM(100)의 포트(108)(도 5)에 어드레스를 제공하며, 버스(218)를 경유하여 제어기(192)에 ID를 제공한다. 또한, 레지스터(196<sub>1</sub>)는 아래에서 설명되는 바와 같이, 어드레스/ID 버스(220<sub>1</sub>)를 경유하여 레지스터(196<sub>2</sub>~196<sub>n</sub>)로부터 어드레스와 ID를 또한 수신한다. 게다가, 레지스터(196<sub>1</sub>)는 라인(222)을 경유하여 제어기(192)로부터 SHIFT/LOAD 신호를 수신한다. 파이프라인 회로(80)(도 5)가 PLIC인 경우에, 버스(216)는 글로벌 라우팅 리소스를 활용하여 전형적으로 형성되고, 버스(220<sub>1</sub>~220<sub>n-1</sub>)와 라인(222)은 로컬 라우팅 리소스를 활용하여 바람직하게 형성된다.

FIFO EMPTY 신호를 수신하고 WRITE/READ FIFO와 SHIFT/LOAD 신호를 발생시키고 읽은 데이터와 대응 ID를 제공하는 것에 더하여, 제어기(192)는 버스(224)를 경유하여 DPSRAM(100)(도 5)의 포트(108)로부터 읽은 데이터를 수신하고 라인(226)상에 READ DPSRAM 신호를 발생시켜, 이 신호를 포트(108)에 결합시킨다. 파이프라인 회로(80)(도 5)가 PLIC인 경우, 버스(224)와 라인(226)은 글로벌 라우팅 리소스를 활용하여 전형적으로 형성된다.

도 7을 참조하여, 인터페이스(104)의 동작을 살펴보면 다음과 같다.

우선, FIFO(194<sub>1</sub>)는 읽은 어드레스와 관련된 FIFO의 현재 상태("풀" 또는 "풀 아님")에 대응하여 로직 레벨로 FIFO FULL 신호를 구동한다. 즉, FIFO(194<sub>1</sub>)는 판독될 어드레스로(풀) 채워진 경우, FIFO FULL의 로직 레벨은 하나의 레벨로 구동시키고, FIFO가 읽혀진 어드레스로 풀 아님(채워지지 않은)의 경우, FIFO FULL의 로직 레벨을 또 다른 레벨로 구동시킨다.

다음으로, FIFO(194<sub>1</sub>)가 판독된 어드레스로 풀 아님이므로 파이프라인(74<sub>1</sub>)이 더 많은 입력 데이터를 처리할 준비가 되어 있는 경우, 파이프라인은 데이터의 어드레스를 구동시켜, 버스(198<sub>1</sub>)에서 읽혀지게 하고, 기록 레벨에 READ/WRITE FIFO 신호를 행사하여, FIFO에 어드레스를 로딩한다. 도 5와 함께 위에서 설명한 바와 같이, 파이프라인(74<sub>1</sub>)은 시퀀스 관리자(148)를 경유하여 입력 데이터 큐(122)로부터 어드레스를 연다. 그러나, FIFO(194<sub>1</sub>)가 판독된 어드레스로 채워진 경우, FIFO가 판독된 데이터를 로딩하기 전에 풀 아님이 때까지 파이프라인(74<sub>1</sub>)은 대기한다.

그 다음, FIFO(194<sub>1</sub>)는 판독된 어드레스에 관련된 FIFO의 현재 상태("엠프티" 또는 "엠프티 아님")에 대응하여 로직 레벨로 FIFO EMPTY 신호를 구동시킨다. 즉, FIFO(194<sub>1</sub>)는 적어도 하나의 임의의 어드레스로 로드되는 경우, FIFO EMPTY의 로직 레벨을 하나의 레벨로 구동시키고, FIFO가 임의의 어드레스로 로드된 경우, FIFO EMPTY의 로직 레벨을 또 다른 레벨로 구동시킨다.

다음으로, FIFO(194<sub>1</sub>)가 엠프티 아님인 경우, 제어기(192)는 WRITE/READ FIFO 신호를 읽기 로직 레벨로 행사하고 SHIFT/LOAD 신호를 로드 로직 레벨로 구동시켜, FIFO에서 레지스터(196<sub>1</sub>)로 첫번째 로드된 어드레스와 ID를 로드한다.

채널(190<sub>2</sub>~190<sub>n</sub>)은 FIFO(194<sub>2</sub>~194<sub>n</sub>)로부터 레지스터(196<sub>2</sub>~196<sub>n</sub>)로 첫 번째 로드된 어드레스와 ID를 각각 로드되는 것과 유사한 방식으로 동작한다. 모든 FIFO(194<sub>2</sub>~194<sub>n</sub>)가 엠프티인 경우 제어기(192)는 진행 전에 어드레스를 수신하기 위하여 적어도 하나의 FIFO를 대기한다.

그 다음, 제어기(192)는 시프트 로직 레벨로 SHIFT/LOAD 신호를 구동하고 READ DPSRAM 신호를 행사하여, 레지스터(196<sub>1</sub>~196<sub>n</sub>)로부터 어드레스와 ID 버스(216, 218)상에 데이터와 어드레스를 시계열적으로 시프트시키고 버스(224)를 경유하여 DPSRAM(100)의 대응 위치에 데이터를 시계열적으로 읽는다.

다음으로, 제어기(192)는 수신한 데이터와 대응 ID - ID는 각 FIFO(194<sub>1</sub>~194<sub>n</sub>)를 데이터를 의도된 수신인지 아닌지를 결정한다-를 버스(212)상에 구동시키고, 쓰기 레벨로 WRITE/READ FIFO 신호를 구동시켜, 각 FIFO(194<sub>1</sub>~194<sub>n</sub>)에 데이터를 시계열적으로 쓴다.

그리하여, 하드웨어인 파이프라인(74<sub>1</sub>~74<sub>n</sub>)은 그들의 READ/WRITE FIFO 신호를 읽기 레벨로 시계열적으로 확증하고 버스(214<sub>1</sub>~214<sub>n</sub>)를 경유하여 데이터를 시계열적으로 읽는다.

도 7을 참조하여 이들 데이터 판독 연산을 상세히 살펴보면 다음과 같다.

첫번째 시프트 사이클 동안에, 제어기(192)는 레지스터(916<sub>1</sub>)로부터 버스(216, 218)상에 어드레스와 ID를 각각 시프트시키고, 읽기 DPSRAM을 행사하여, 버스(224)를 경유하여 DPSRAM(100)의 대응 위치로부터 데이터를 읽고 버스(218)로부터 ID를 읽는다. 다음으로, 제어기(192)는 쓰기 레벨로 라인(208<sub>1</sub>)상에 WRITE/READ FIFO 신호를 구동시키고 버스(212)상에 수신된 데이터와 ID를 구동시킨다. ID는 FIFO(194<sub>1</sub>)로부터 ID이기 때문에, FIFO(194<sub>1</sub>)는 ID를 인식하여 WRITE/READ FIFO 신호에 응답하여 버스(212)로부터 데이터를 로드한다. 버스(212)상에 ID가 그 ID들에 대응하지 않기 때문에 나머지 FIFO(194<sub>2</sub>~194<sub>n</sub>)는 데이터를 로드하지 않는다. 그 다음, 파이프라인(74<sub>1</sub>)은 읽기 레벨로 라인(202<sub>1</sub>)상에 READ/WRITE FIFO 신호를 행사하고 버스(214<sub>1</sub>)를 경유하여 읽은 데이터를 회수한다. 또한, 첫 번째 시프트 사이클 동안에, 레지스터(196<sub>2</sub>)로부터 어드레스와 ID는 레지스터(196<sub>2</sub>)로 시프트되고, 레지스터(196<sub>2</sub>)(도시하지 않음)로부터 어드레스와 ID는 레지스터로 시프트되고 나머지도 같다. 다른 예로서, 제어기(192)는 ID를 인식하고 쓰기 레벨로 라인(208<sub>1</sub>)상에 WRITE/READ FIFO 신호만을 구동시킬 수 있다. 이는 제어기(192)의 필요성을 제거하여 ID를 FIFO(194<sub>1</sub>~194<sub>n</sub>)에 보낸다. 또 다른 예로서, WRITE/READ FIFO 신호는 읽기 신호만을 수 있고, 버스(212)상에 ID가 FIFO(194<sub>1</sub>)의 ID를 정합시키는 경우 FIFO(194<sub>1</sub>)(뿐만 아니라 다른 FIFO(194<sub>2</sub>~194<sub>n</sub>)도) 버스(212)상에 데이터를 로드할 수 있다. 이는 제어기(192)의 필요성을 제거하여 쓰기 신호를 발생시킨다.

두번째 시프트 사이클 동안에, 레지스터(196<sub>1</sub>)로부터 어드레스와 ID는 버스(216, 218)상으로 이동되어 제어기(192)는 FIFO(194<sub>2</sub>)에 의하여 특정된 DPSRAM(100)의 위치로부터 데이터를 읽는다. 다음으로, 제어기(192)는 WRITE/READ FIFO 신호를 쓰기 레벨로 구동시키고 버스(212)상에 수신된 데이터와 ID를 구동시킨다. ID는 FIFO(194<sub>2</sub>)로부터 ID이므로, FIFO(194<sub>2</sub>)는 ID를 인식하여 버스(212)로부터 데이터를 로드한다. 버스(212)상의 ID는 이들 ID와 대응하지 않기 때문에 나머지 FIFO(194<sub>1</sub> 및 194<sub>3</sub>~194<sub>n</sub>)는 데이터를 로드하지 않는다. 버스(212)상의 ID는 그 ID와 대응하지 않기 때문에 나머지 FIFO(194<sub>1</sub> 및 194<sub>3</sub>~194<sub>n</sub>)는 데이터를 로드하지 않는다. 그 다음, 파이프라인(74<sub>2</sub>)은 그 READ/WRITE FIFO 신호

호를 읽기 레벨로 행사하고 버스(214<sub>2</sub>)를 경유하여 읽기 데이터를 회수한다. 두 번째 시프트 사이클 동안에, 레지스터(196<sub>2</sub>)로부터 어드레스와 ID는 레지스터(196<sub>1</sub>)에 시프트되어 레지스터(196<sub>3</sub>)(도시하지 않음)로부터 어드레스와 ID는 레지스터(196<sub>2</sub>)에 시프트되고 나머지도 동일하다.

(FIFO(194<sub>n</sub>)로부터 어드레스와 ID인) 레지스터(196<sub>n</sub>)로부터 어드레스와 ID가 버스에 각각 시프트될 때까지 n 시프트 사이클은 계속된다. 제어기(192)는 SHIFT/LOAD 신호를 펄스하거나 레지스터(196<sub>1</sub>~196<sub>n</sub>)에 결합된 시프트 클럭 신호(도시하지 않음)를 발생하여 이들 시프트 사이클을 수행할 수 있다. 더욱이, 대응 FIFO(194<sub>1</sub>~194<sub>n</sub>)가 애플티이기 때문에, 레지스터(196<sub>1</sub>~196<sub>2</sub>)중 하나가 특정 시프트 동작 동안에 애플티인 경우, 제어기(192)는 애플티 레지스터를 바이패스할 수 있어 버스(216)상에 영 어드레스를 시프팅하는 것을 피하여 시프트 동작을 단축시킨다.

도 5와 도 6을 참조하여, 본 발명의 일실시예를 따르면, 인터페이스(144)는 인터페이스(140)와 유사하고, 인터페이스(136)는 하나의 읽기 채널(190)만을 포함하고 ID 회로를 포함하지 않는 것을 제외하고는 인터페이스(140)와 또한 유사하다.

도 8은 본 발명의 일실시예에 따른 도4의 파이프라인 유닛(230)의 개략적인 블록도이다. 파이프라인 유닛(230)은 다중 파이프라인 회로(80)~ 여기서는 두개의 파이프라인 회로(80a 및 80b)를 포함하는 것을 제외하고는 도4의 파이프라인 유닛(78)과 유사하다. 파이프라인 회로(80)의 숫자를 증가시키는 것은 전형적으로 하드와이어드 파이프라인(74<sub>1</sub>~74<sub>n</sub>)의 숫자 n의 증가를 허용하여 파이프라인 유닛(78)과 비교하여 파이프라인 유닛(230)의 기능을 증가시킨다.

도 8의 파이프라인 유닛(230)에서, 서비스 구성요소들, 즉, 통신 인터페이스(82), 제어기(86), 예외 관리자(88), 구성 관리자(90), 및 선택적 산업-표준 버스 인터페이스(91)는 파이프라인 회로(80a)상에 배치되고, 파이프라인(74<sub>1</sub>~74<sub>n</sub>)과 통신 셀(84)은 파이프라인 회로(80b)상에 배치된다. 분리된 파이프라인 회로상에 파이프라인(74<sub>1</sub>~74<sub>n</sub>)과 서비스 구성요소들을 위치시켜, 동일 파이프라인 회로상에 서비스 구성요소와 파이프라인을 위치시키는 것보다 더 많은 숫자의 파이프라인 및/또는 더 많은 복잡한 파이프라인을 포함시킬 수 있다. 이와 달리, 파이프라인(74<sub>1</sub>~74<sub>n</sub>)을 인터페이스(82)와 제어기(86)에 인터페이스하는 통신 셀(84) 부분은 파이프라인 회로(80a)상에 배치될 수 있다.

도 9는 본 발명의 일실시예에 따른 도8의 파이프라인 유닛(230)의 파이프라인 회로(80a 및 80b)와 데이터 메모리(92)의 개략적인 블록도이다. 파이프라인 구성요소가 두 개의 파이프라인 회로상에 배치되는 것외에는 도 9의 파이프라인 회로(80a 및 80b)와 메모리의 구조와 동작은 도 5의 파이프라인 회로(80) 및 메모리(92)와 동일하다.

앞의 설명으로부터 당업자는 본 발명을 실시하거나 활용할 수 있다. 본 발명의 실시예들을 당업자는 다양하게 변형시킬 수 있다는 것은 분명하고, 본 발명의 요지와 범위를 벗어나지 않고 다른 실시예 및 응용 분야에 적용할 수 있다. 따라서, 본 발명의 실시예들은 본 발명을 제한하기 위한 것이 아니고 여기에서 공개한 원리와 특징에 넓게 해석되어야 할 것이다.

(57) 청구의 범위

## 청구항 1.

메모리; 및

상기 메모리와 결합되어 있으면서,

데이터를 수신하고,

상기 데이터를 상기 메모리에 로드하고,

상기 메모리로부터 상기 데이터를 검색하고,

상기 검색된 데이터를 처리하고, 그리고

상기 처리된 데이터를 외부 소스로 제공하도록 동작 가능한 하드와이어드~파이프라인 회로를 구비하는 것을 특징으로 하는 파이프라인 가속기.

## 청구항 2.

청구항 1에 있어서,

상기 메모리는 제1 집적회로상에 배치되어 있고, 상기 파이프라인 회로는 제2 집적회로상에 배치되어 있는 것을 특징으로 하는 파이프라인 가속기.

## 청구항 3.

청구항 1에 있어서,

상기 파이프라인 회로는 펠드~프로그램가능한 게이트 어레이상에 배치되어 있는 것을 특징으로 하는 파이프라인 가속기.

## 청구항 4.

청구항 1에 있어서,

상기 파이프라인 회로는,

상기 처리된 데이터를 상기 메모리에 로드하고;

상기 메모리로부터 상기 처리된 데이터를 검색하고; 그리고

상기 검색된 처리된 데이터를 상기 외부 소스로 제공함으로써 상기 처리된 데이터를 상기 외부 소스로 제공하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

## 청구항 5.

청구항 1에 있어서,

상기 외부 소스는 프로세서를 구비하고, 그리고

상기 파이프라인 회로는 상기 프로세서로부터 상기 데이터를 수신하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

## 청구항 6.

프로세서; 및

상기 프로세서와 결합되어 있는 파이프라인 가속기를 구비하며,

상기 파이프라인 가속기는,

메모리; 및

상기 메모리와 결합되어 있고,

상기 프로세서로부터 데이터를 수신하고,

상기 데이터를 상기 메모리에 로드하고,

상기 메모리로부터 상기 데이터를 검색하고,

상기 검색된 데이터를 처리하고, 그리고

상기 처리된 데이터를 상기 프로세서로 제공하도록 동작 가능한 하드와이어드-파이프라인 회로를 구비하는 것을 특징으로 하는 컴퓨팅 머신.

## 청구항 7.

메모리; 및

상기 메모리와 결합되어 있으며,

데이터를 수신하고,

상기 수신된 데이터를 처리하고,

상기 처리된 데이터를 상기 메모리에 로드하고,

상기 메모리로부터 상기 처리된 데이터를 검색하고, 그리고

상기 검색된 처리된 데이터를 외부 소스로 제공할도록 동작 가능한 하드와이어드-파이프라인 회로를 구비하는 것을 특징으로 하는 파이프라인 가속기.

## 청구항 8.

프로세서; 및

상기 프로세서와 결합된 파이프라인 가속기를 구비하며,

상기 파이프라인 가속기는,

메모리; 및

상기 메모리와 결합되어 있으며,

상기 프로세서로부터 데이터를 수신하고,

상기 수신된 데이터를 처리하고,

상기 처리된 데이터를 상기 메모리에 로드하고,



상기 메모리로부터 상기 처리된 데이터를 검색하고, 그리고

상기 검색된 처리된 데이터를 상기 프로세서로 제공하도록 동작 가능한 하드와이어드-파이프라인 회로를 구비하는 것을 특징으로 하는 컴퓨터 시스템.

## 청구항 9.

제1 및 제2 메모리; 및

상기 제1 및 제2 메모리와 결합되어 있으며,

외부 소스로부터 원 데이터를 수신하고 상기 원 데이터를 상기 제1 메모리에 로드하도록 동작 가능한 입력-데이터 처리기,

상기 원 데이터를 처리하도록 동작 가능한 하드와이어드 파이프라인,

상기 제1 메모리로부터 상기 원 데이터를 검색하고, 상기 검색된 원 데이터를 상기 하드와이어드 파이프라인에 제공하고, 상기 하드와이어드 파이프라인으로부터 상기 처리된 데이터를 상기 제2 메모리에 로드하도록 동작 가능한 파이프라인 인터페이스, 및

상기 제2 메모리로부터 상기 처리된 데이터를 검색하고 상기 처리된 데이터를 상기 외부 소스로 제공하도록 동작 가능한 출력-데이터 처리기를 구비하는 하드와이어드-파이프라인 회로를 구비하는 것을 특징으로 하는 파이프라인 가속기.

## 청구항 10.

청구항 9에 있어서,

상기 제1 및 제2 메모리 각각에는 각각의 제1 및 제2 포트가 포함되어 있고,

상기 입력-데이터 처리기는 상기 제1 메모리의 상기 제1 포트를 통해 상기 원 데이터를 로드하도록 동작 가능하며,

상기 파이프라인 인터페이스는 상기 제1 메모리의 상기 제2 포트를 통해 상기 원 데이터를 검색하고 상기 처리된 데이터를 상기 제2 메모리의 상기 제1 포트를 통해 로드하도록 동작 가능하며,

상기 출력-데이터 처리기는 상기 제2 메모리의 상기 제2 포트를 통해 상기 처리된 데이터를 검색하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

## 청구항 11.

청구항 9에 있어서,

상기 하드와이어드-파이프라인 회로에 결합된 제3 메모리를 더 구비하고,

상기 하드와이어드 파이프라인은 상기 원 데이터를 처리하는 동안 중간 데이터를 발생하도록 동작 가능하며;

상기 파이프라인 인터페이스는 상기 제3 메모리로부터 상기 중간 데이터를 로드하고 상기 제3 메모리로부터 상기 중간 데이터를 검색하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

**청구항 12.**

청구항 9에 있어서,

상기 제1 및 제2 메모리는 각각 제1 및 제2 집적회로상에 배치되어 있으며,

상기 파이프라인 회로는 펄스-프로그램가능한 게이트 어레이상에 배치되어 있는 것을 특징으로 하는 파이프라인 가속기.

**청구항 13.**

청구항 9에 있어서,

상기 입력-데이터 처리기 및 상기 파이프라인 인터페이스와 결합된 출력-데이터 큐를 더 구비하고,

상기 입력-데이터 처리기는 상기 제1 메모리 내부의 상기 원 데이터의 위치로 상기 입력-데이터 큐에 포인터를 로드하도록 동작 가능하며; 그리고

상기 파이프라인 인터페이스는 상기 포인터를 이용하여 상기 위치로부터 상기 원 데이터를 검색하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

**청구항 14.**

청구항 9에 있어서,

상기 출력-데이터 처리기 및 상기 파이프라인 인터페이스와 결합된 출력-데이터 큐를 더 구비하고,

상기 파이프라인 인터페이스는 상기 제2 메모리 내부의 상기 처리된 데이터의 위치로 상기 출력-데이터 큐에 포인터를 로드하도록 동작 가능하며; 그리고

상기 출력-데이터 처리기는 상기 포인터를 사용하여 상기 위치로부터 상기 원 데이터를 검색하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

**청구항 15.**

청구항 9에 있어서,

상기 입력-데이터 처리기, 하드와이어드 파이프라인, 파이프라인 인터페이스 및 출력-데이터 처리기 각각은 개별적인 동작 구성을 가지며,

상기 입력-데이터 처리기, 하드와이어드 파이프라인, 파이프라인 인터페이스 및 출력-데이터 처리기의 상기 동작 구성과 결합 및 이를 설정하도록 동작 가능한 구성 관리자를 더 구비하는 것을 특징으로 하는 파이프라인 가속기.

**청구항 16.**

청구항 9에 있어서,

상기 입력-데이터 처리기, 하드와이어드 파이프라인, 파이프라인 인터페이스 및 데이터-출력 처리기 각각은 개별적인 동작 상태를 가지며;

상기 동작 상태에 응답하여 상기 입력-데이터 처리기, 하드와이어드 파이프라인, 파이프라인 인터페이스 또는 출력-데이터 처리기와 결합 및 이들에서의 예외를 식별하도록 동작가능한 예외 관리자를 더 구비하는 것을 특징으로 하는 파이프라인 가속기.

#### 청구항 17.

데이터를 처리 하도록 동작 가능한 하드와이어드 파이프라인; 및

상기 하드와이어드 파이프라인과 결합되고,

상기 데이터를 수신하고,

상기 데이터가 상기 하드와이어드 파이프라인을 향한 것인지를 결정하고,

상기 데이터가 상기 하드와이어드 파이프라인을 향한 것인 경우 상기 데이터를 상기 하드와이어드 파이프라인으로 제공하도록 동작 가능한 입력-데이터 처리기를 구비하는 것을 특징으로 하는 파이프라인 가속기.

#### 청구항 18.

청구항 17에 있어서,

상기 입력-데이터 처리기는,

헤더 및 상기 데이터를 포함하는 메시지를 수신하고, 상기 메시지에서부터 상기 데이터를 추출하는 것에 의해 상기 데이터를 수신하고,

상기 헤더를 분석하여 상기 데이터가 상기 하드와이어드 파이프라인을 향한 것인지를 결정하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

#### 청구항 19.

청구항 17에 있어서,

상기 하드와이어드 파이프라인 및 상기 입력-데이터 처리기는 하나의 월드-프로그램가능한 게이트 어레이상에 배치되는 것을 특징으로 하는 파이프라인 가속기.

#### 청구항 20.

청구항 17에 있어서,

상기 하드와이어드 파이프라인 및 상기 입력-데이터 처리기는 각각의 월드-프로그램가능한 게이트 어레이상에 배치되는 것을 특징으로 하는 파이프라인 가속기.

## 청구항 21.

프로세서; 및

상기 프로세서와 결합된 파이프라인 가속기를 구비하고,

상기 파이프라인 가속기는,

데이터 처리 가능한 하드와이어드 파이프라인, 및

상기 하드와이어드 파이프라인과 결합되고,

상기 프로세서로부터 상기 데이터를 수신하고,

상기 데이터가 상기 하드와이어드 파이프라인을 향한 것인지를 결정하고,

상기 데이터가 상기 하드와이어드 파이프라인을 향한 것인 경우에 상기 데이터를 상기 하드와이어드 파이프라인에 제공하도록 동작 가능한 입력-데이터 처리기를 구비하는 것을 특징으로 하는 컴퓨팅 머신.

## 청구항 22.

데이터를 발생하도록 동작 가능한 하드와이어드 파이프라인; 및

상기 하드와이어드 파이프라인과 결합되어 있으며,

상기 데이터를 수신하고,

상기 데이터의 목적지를 결정하고,

상기 목적지로 상기 데이터를 제공하도록 동작 가능한 출력-데이터 처리기를 구비하는 것을 특징으로 하는 파이프라인 가속기.

## 청구항 23.

청구항 22에 있어서,

상기 출력-데이터 처리기는,

상기 데이터의 타입을 식별하고, 상기 데이터의 타입에 기초하여 상기 목적지를 결정함으로써 상기 데이터의 목적지를 결정하고,

상기 목적지를 식별하고 상기 데이터를 포함하는 메시지를 발생하고, 상기 메시지를 상기 목적지로 제공함으로써 상기 데이터를 상기 목적지로 제공하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

## 청구항 24.

애플리케이션의 스레드를 실행하도록 동작 가능한 프로세서; 및

상기 프로세서와 결합되어 있으며,

데이터를 발생하도록 동작 가능한 하드와이어드 파이프라인, 및

상기 하드와이어드 파이프라인과 결합되어 있으며,

상기 데이터를 수신하고,

상기 데이터를 서브스크라이브하는 상기 애플리케이션의 스레드를 식별하고,

상기 서브스크라이브 스레드로 상기 데이터를 제공하도록 동작 가능한 출력-데이터 처리기를 구비하는 파이프라인 가속기를 구비하는 것을 특징으로 하는 컴퓨팅 머신.

## 청구항 25.

데이터 값을 처리하도록 동작 가능한 하드와이어드 파이프라인; 및

상기 하드와이어드 파이프라인에 결합되어 있고 및 이것의 동작을 제어하도록 동작 가능한 시퀀스 관리자를 구비하는 것을 특징으로 하는 파이프라인 가속기.

## 청구항 26.

청구항 25에 있어서,

상기 시퀀스 관리자는 상기 하드와이어드 파이프라인이 상기 데이터 값을 수신하는 순서를 제어하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

## 청구항 27.

청구항 25에 있어서,

상기 시퀀스 관리자는,

이벤트를 수신하고,

상기 이벤트에 응답하여 상기 하드와이어드 파이프라인을 제어하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

## 청구항 28.

청구항 25에 있어서,

상기 시퀀스 관리자는,

동기화 신호를 수신하고,

상기 동기화 신호에 응답하여 상기 하드와이어드 파이프라인의 동작을 제어하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

## 청구항 29.

청구항 25에 있어서,

상기 시스템 관리자는,

상기 하드웨어어드 파이프라인에 관련된 출현을 감지하고,

상기 출현에 응답하여 이벤트를 발생하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

## 청구항 30.

데이터 및 이벤트를 발생하도록 동작 가능한 프로세서; 및

상기 프로세서와 결합되어 있으며,

상기 프로세서로부터 상기 데이터를 수신하고 상기 수신된 데이터를 처리하도록 동작 가능한 하드웨어어드 파이프라인; 및

상기 하드웨어어드 파이프라인에 결합되어 있고 상기 프로세서로부터 상기 이벤트를 수신하고 상기 이벤트에 응답하여 상기 하드웨어어드 파이프라인의 동작을 제어 하도록 동작 가능한 시스템 관리자를 구비하는 파이프라인 가속기를 구비하는 것을 특징으로 하는 컴퓨터 시스템.

## 청구항 31.

동작 구성을 가지며 데이터를 처리하도록 동작 가능한 하드웨어어드-파이프라인 회로; 및

상기 하드웨어어드-파이프라인 회로에 결합되어 있고 상기 동작 구성을 설정하도록 동작 가능한 구성 관리자를 구비하는 것을 특징으로 하는 파이프라인 가속기.

## 청구항 32.

청구항 31에 있어서,

상기 하드웨어어드-파이프라인 회로에는 구성 레지스터가 포함되어 있고,

상기 구성 관리자는 구성 값을 상기 구성 레지스터에 로드함으로써 상기 동작 구성을 설정하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

## 청구항 33.

청구항 32에 있어서,

상기 구성 관리자는 외부 소스에서 상기 구성 값을 수신하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

## 청구항 34.

데이터 및 구성 값을 발생하도록 동작 가능한 프로세서; 및

상기 프로세서와 결합되어 있으며,

동작 구성을 가지고 있고 상기 데이터를 처리하도록 동작 가능한 하드웨어드-파이프라인 회로 및

상기 하드웨어드-파이프라인 회로에 결합되어 있고 상기 구성 값에 응답하여 상기 동작 구성을 설정하도록 동작 가능한 구성 관리자를 구비하는 파이프라인 가속기를 구비하는 것을 특징으로 하는 컴퓨팅 머신.

### 청구항 35.

동작 상태를 가지며 데이터를 처리하도록 동작 가능한 하드웨어드-파이프라인 회로; 및

상기 하드웨어드-파이프라인 회로에 결합되어 있고 상기 동작 상태에 응답하여 상기 하드웨어드-파이프라인 회로의 동작 상태내의 예외를 식별하도록 동작 가능한 예외 관리자를 구비하는 것을 특징으로 하는 파이프라인 가속기.

### 청구항 36.

청구항 35에 있어서,

상기 하드웨어드-파이프라인 회로는 상기 동작 상태를 나타내는 상태값을 발생하도록 동작 가능하며,

상기 예외 관리자는 상기 상태 값에 응답하여 상기 예외를 식별하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

### 청구항 37.

청구항 36에 있어서,

상기 하드웨어드-파이프라인 회로에는 상기 상태 값을 저장하도록 동작 가능한 상태 레지스터가 포함되어 있으며,

상기 예외 관리자는 상기 상태 레지스터로부터 상기 상태 값을 수신하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

### 청구항 38.

청구항 35에 있어서,

상기 예외 관리자는 상기 하드웨어드-파이프라인 회로의 동작 상태 내의 예외를 외부 소스로 식별하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

### 청구항 39.

데이터를 발생하도록 동작 가능한 프로세서; 및

파이프라인 가속기를 구비하고,

상기 파이프라인 가속기는,

동작 상태를 가지며 데이터를 처리하고 상기 동작 상태를 나타내는 상태값을 발생하도록 동작 가능한 하드와이어드-파이프라인 회로, 및

상기 하드와이어드-파이프라인 회로에 결합되어 있으며 상기 상태값에 응답하여 상기 하드와이어드-파이프라인 회로의 상기 동작 상태 내의 예외를 식별하고 상기 예외의 프로세서를 통보하도록 동작 가능한 것을 특징으로 하는 예외 관리자를 구비하는 것을 특징으로 하는 컴퓨팅 머신.

#### 청구항 40.

동작 상태를 가지며 데이터를 처리하도록 동작 가능한 하드와이어드-파이프라인 회로, 및 상기 하드와이어드-파이프라인 회로와 결합되어 있고 상기 동작 상태를 나타내는 상태값을 발생하도록 동작 가능한 예외 관리자를 구비하는 파이프라인 가속기; 및

상기 파이프라인 가속기와 결합되어 있으며 상기 데이터의 발생, 상기 상태값의 수신 및 상기 상태값의 분석에 의해 상기 하드와이어드-파이프라인 회로의 장애를 결정하도록 동작 가능한 프로세서를 구비하는 것을 특징으로 하는 컴퓨팅 머신.

#### 청구항 41.

메모리에 데이터를 로드하는 단계;

상기 메모리로부터 상기 데이터를 검색하는 단계;

하드와이어드-파이프라인 회로로 상기 검색된 데이터를 처리하는 단계; 및

상기 처리된 데이터를 외부 소스로 제공하는 단계를 구비하는 것을 특징으로 하는 방법.

#### 청구항 42.

청구항 41에 있어서,

상기 처리된 데이터를 제공하는 단계는;

상기 처리된 데이터를 상기 메모리로 로드하는 단계;

상기 메모리로부터 상기 처리된 데이터를 검색하는 단계; 및

상기 검색된 처리된 데이터를 상기 외부 소스로 제공하는 단계를 구비하는 것을 특징으로 하는 방법.

#### 청구항 43.

하드와이어드-파이프라인 회로로 데이터를 처리하는 단계;

상기 처리된 데이터를 메모리에 로드하는 단계;

상기 메모리로부터 상기 처리된 데이터를 검색하는 단계; 및



상기 검색된 처리된 데이터를 외부 소스로 제공하는 단계를 구비하는 것을 특징으로 하는 방법.

#### 청구항 44.

외부 소스로부터의 원 데이터를 제1 메모리에 로드하는 단계;

상기 제1 메모리로부터 상기 원 데이터를 검색하는 단계;

하드웨어드 파이프라인으로 상기 검색된 데이터를 처리하는 단계;

상기 하드웨어드 파이프라인으로부터 상기 처리된 데이터를 제2 메모리에 로드하는 단계; 및

상기 제2 메모리로부터 상기 처리된 데이터를 상기 외부 소스로 제공하는 단계를 구비하는 것을 특징으로 하는 방법.

#### 청구항 45.

청구항 44에 있어서,

상기 원 데이터를 로드하는 단계는,

상기 제1 메모리의 제1 포트를 통해 상기 원 데이터를 로드하는 단계;

상기 제1 메모리의 제2 포트를 통해 상기 원 데이터를 검색하는 단계를 포함하는 상기 원 데이터를 검색하는 단계;

상기 제2 메모리의 제1 포트를 통해 상기 처리된 데이터를 로드하는 단계를 포함하는 상기 처리된 데이터를 로드하는 단계; 및

상기 제2 메모리의 제2 포트를 통해 상기 처리된 데이터를 검색하는 단계를 포함하는 상기 처리된 데이터를 제공하는 단계를 구비하는 것을 특징으로 하는 방법.

#### 청구항 46.

청구항 44에 있어서,

상기 원 데이터 처리에 응답하여 상기 하드웨어드 파이프라인으로 중간 데이터를 발생시키는 단계;

상기 중간 데이터를 제3 메모리에 로드하는 단계; 및

상기 제3 메모리로부터 상기 중간 데이터를 상기 하드웨어드 파이프라인으로 되돌려보내는 단계를 더 구비하는 것을 특징으로 하는 방법.

#### 청구항 47.

청구항 44에 있어서,

상기 제1 메모리 내부의 상기 원 데이터의 위치에 입력-메시지 큐로 포인터를 코드하는 단계를 더 구비하고,

상기 원 데이터를 검색하는 단계는 상기 포인터를 이용하여 상기 위치로부터 상기 원 데이터를 검색하는 단계를 포함하는 것을 특징으로 하는 방법.

#### 청구항 48.

청구항 44에 있어서,

상기 제2 메모리 내부의 상기 처리된 데이터의 위치에 출력-메시지 큐로 포인터를 로드하는 단계를 더 구비하고,

상기 처리된 데이터를 검색하는 단계는 상기 포인터를 이용하여 상기 위치로부터 상기 처리된 데이터를 검색하는 단계를 포함하는 것을 특징으로 하는 방법.

#### 청구항 49.

청구항 44에 있어서,

상기 원 데이터를 로드 및 검색하고, 상기 검색된 데이터의 처리, 및 상기 처리된 데이터를 로드 및 제공하기 위한 파라미터를 설정하는 단계를 더 구비하는 것을 특징으로 하는 방법.

#### 청구항 50.

청구항 44에 있어서,

상기 원 데이터의 로드 및 검색, 상기 검색된 데이터의 처리, 및 상기 처리된 데이터의 로드 및 제공 동안에 에러의 발생을 결정하는 단계를 더 구비하는 것을 특징으로 하는 방법.

#### 청구항 51.

데이터를 수신하는 단계;

상기 데이터가 하드와이어드 파이프라인을 향한 것인지를 결정하는 단계; 및

상기 데이터가 상기 하드와이어드 파이프라인을 향한 것인 경우 상기 데이터를 상기 하드와이어드 파이프라인에 제공하는 단계를 구비하는 것을 특징으로 하는 방법.

#### 청구항 52.

청구항 51에 있어서,

상기 데이터를 수신하는 단계는:

헤더와 상기 데이터를 포함하는 메시지를 수신하는 단계, 및 상기 메시지에서 상기 데이터를 추출하는 단계를 구비하고,

상기 데이터가 상기 하드와이어드 파이프라인을 향하는 것인지를 결정하는 단계는 상기 헤더를 분석하는 단계를 포함하는 것을 특징으로 하는 방법.

### 청구항 53.

하드와이어드 파이프라인을 가지고 데이터를 발생하는 단계;

상기 데이터의 목적지를 결정하는 단계; 및

상기 데이터를 상기 목적으로 제공하는 단계를 구비하는 것을 특징으로 하는 방법.

### 청구항 54.

청구항 53에 있어서,

상기 데이터의 상기 목적지를 결정하는 단계는,

상기 데이터의 타입을 식별하는 단계, 및 상기 데이터의 상기 타입에 기초하여 상기 목적지를 결정하는 단계를 구비하고,

상기 목적으로 상기 데이터를 제공하는 단계는,

상기 목적지를 식별하고 상기 데이터를 포함하는 메시지를 발생하는 단계, 및 상기 메시지를 상기 목적으로 제공하는 단계를 구비하는 것을 특징으로 하는 방법.

### 청구항 55.

하드와이어드 파이프라인을 가지고 데이터 값을 처리하는 단계; 및

상기 하드와이어드 파이프라인의 동작을 시퀀싱하는 단계를 구비하는 것을 특징으로 하는 방법.

### 청구항 56.

청구항 55에 있어서,

상기 동작을 시퀀싱하는 단계는, 상기 하드와이어드 파이프라인이 상기 데이터 값을 처리하는 순서를 시퀀싱하는 단계를 포함하는 것을 특징으로 하는 방법.

### 청구항 57.

청구항 55에 있어서,

상기 동작을 시퀀싱하는 단계는, 상기 하드와이어드 파이프라인의 동작을 동기화 신호로 동기화하는 단계를 포함하는 것을 특징으로 하는 방법.

### 청구항 58.

청구항 55에 있어서,

상기 하드와이어드 파이프라인의 동작 동안 미리 정의된 출현을 감지하는 단계; 및

상기 출현에 응답하여 이벤트를 발생하는 단계를 포함하는 것을 특징으로 하는 방법.

#### 청구항 59.

구성값을 레지스터에 로드하는 단계; 및

상기 구성값을 가지고 하드와이어드 파이프라인의 동작 구성을 설정하는 단계를 구비하는 것을 특징으로 하는 방법.

#### 청구항 60.

하드와이어드 파이프라인을 가지고 데이터를 처리하는 단계; 및

상기 하드와이어드 파이프라인의 동작 상태를 분석함으로써 상기 처리된 데이터 내의 에러를 식별하는 단계를 구비하는 것을 특징으로 하는 방법.

#### 청구항 61.

라이브러리로부터 통신 인터페이스의 제1 데이터 표현을 검색하는 단계;

상기 통신 인터페이스와 결합될 하드와이어드 파이프라인의 제2 데이터 표현을 발생하는 단계; 및

상기 제1 및 제2 데이터 표현을 결합하여 상기 하드와이어드-파이프라인 회로를 위한 하드-구성 데이터를 발생하는 단계를 구비하는 것을 특징으로 하는 하드와이어드-파이프라인 회로를 디자인 하는 방법.

#### 청구항 62.

청구항 61에 있어서,

상기 제1 및 제2 데이터 표현을 결합하기 전에 상기 서비스 계층의 미리결정된 파라미터를 위한 값을 선택함으로써 상기 제1 데이터 표현을 변형하는 단계를 더 구비하는 것을 특징으로 하는 방법.

#### 청구항 63.

청구항 61에 있어서,

상기 통신 인터페이스는 상기 하드와이어드-회로가 다른 회로와 통신하도록 동작 가능한 것을 특징으로 하는 방법.

#### 청구항 64.

청구항 61에 있어서,

상기 제1 및 제2 데이터 표현을 결합하는 단계는, 상기 제1 및 제2 데이터 표현을 상기 하드-구성 데이터로 컴파일링하는 단계를 구비하는 것을 특징으로 하는 방법.

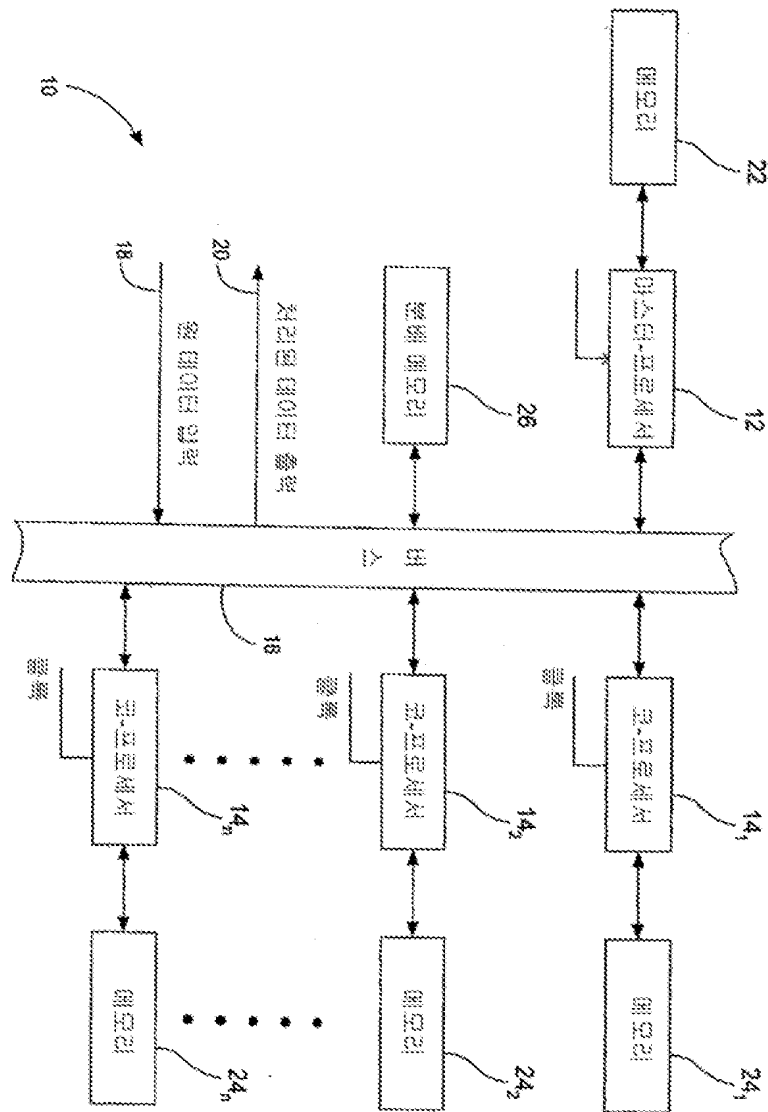
청구항 65.

청구항 61에 있어서,

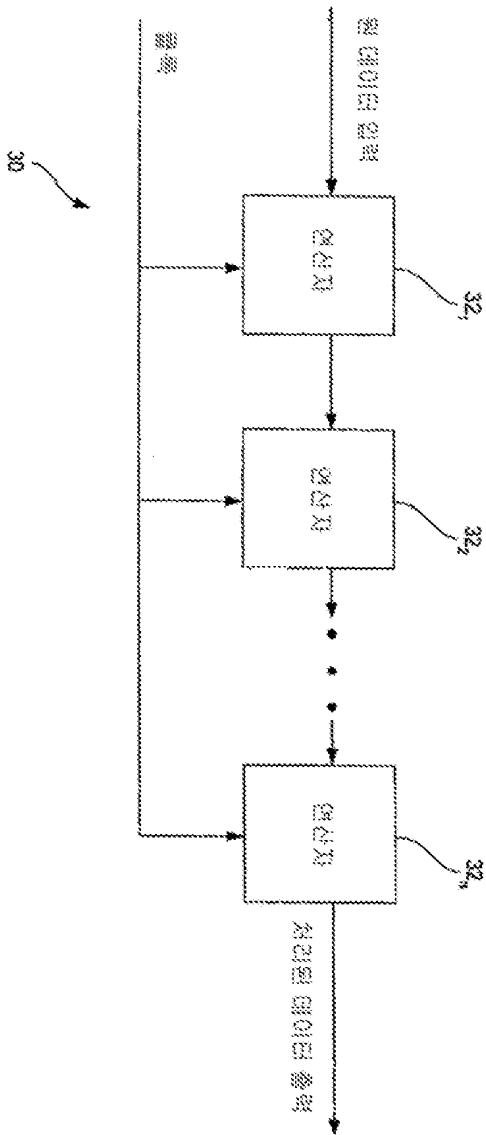
상기 하드-구설 데이터는 평행어를 구비하는 것을 특징으로 하는 방법.

도면

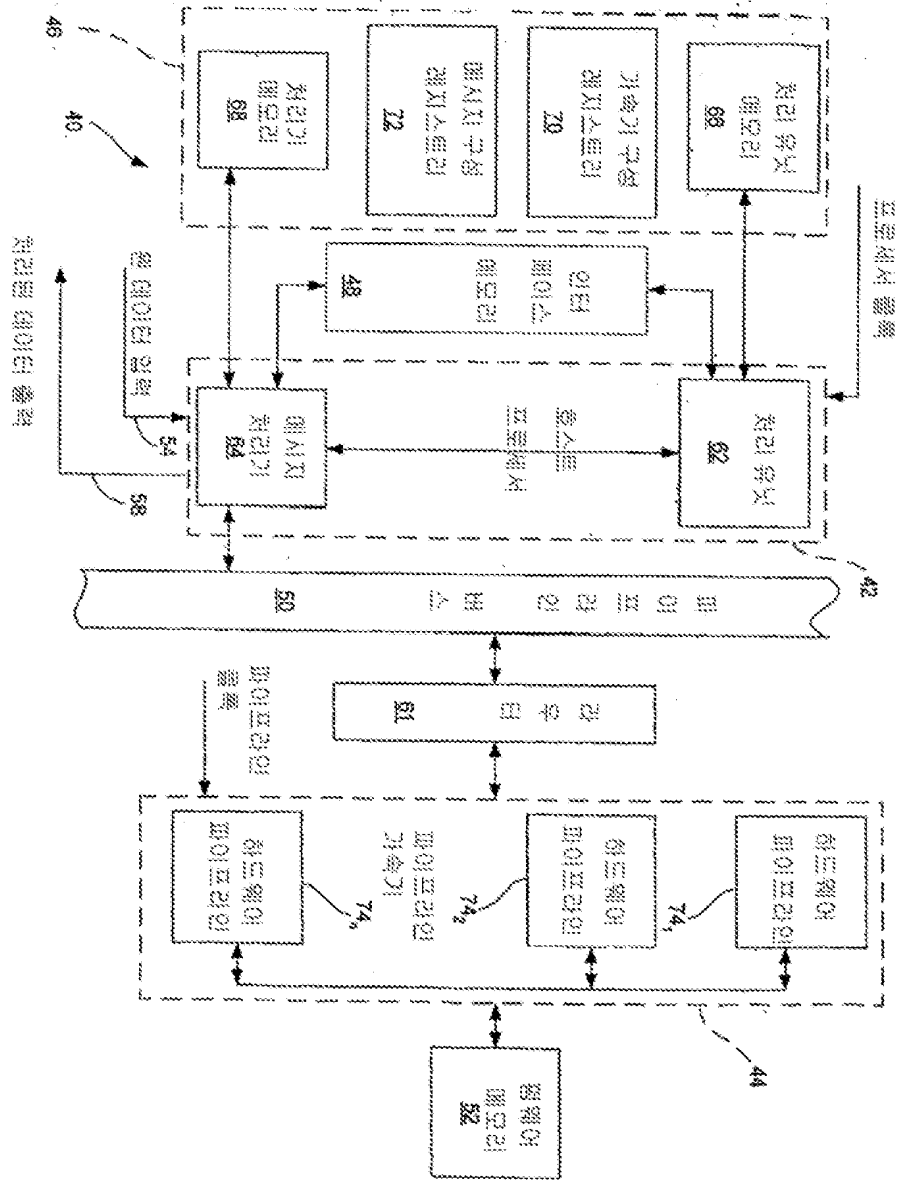
도면1



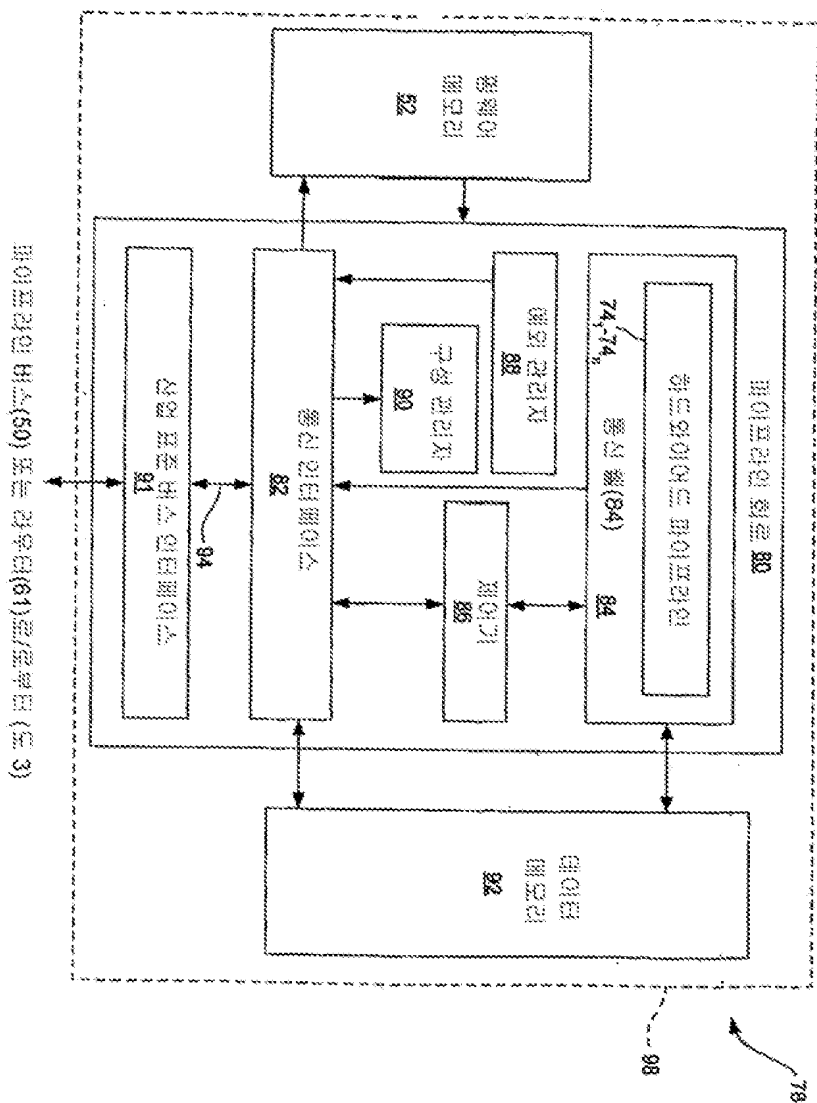
도면2



도면3

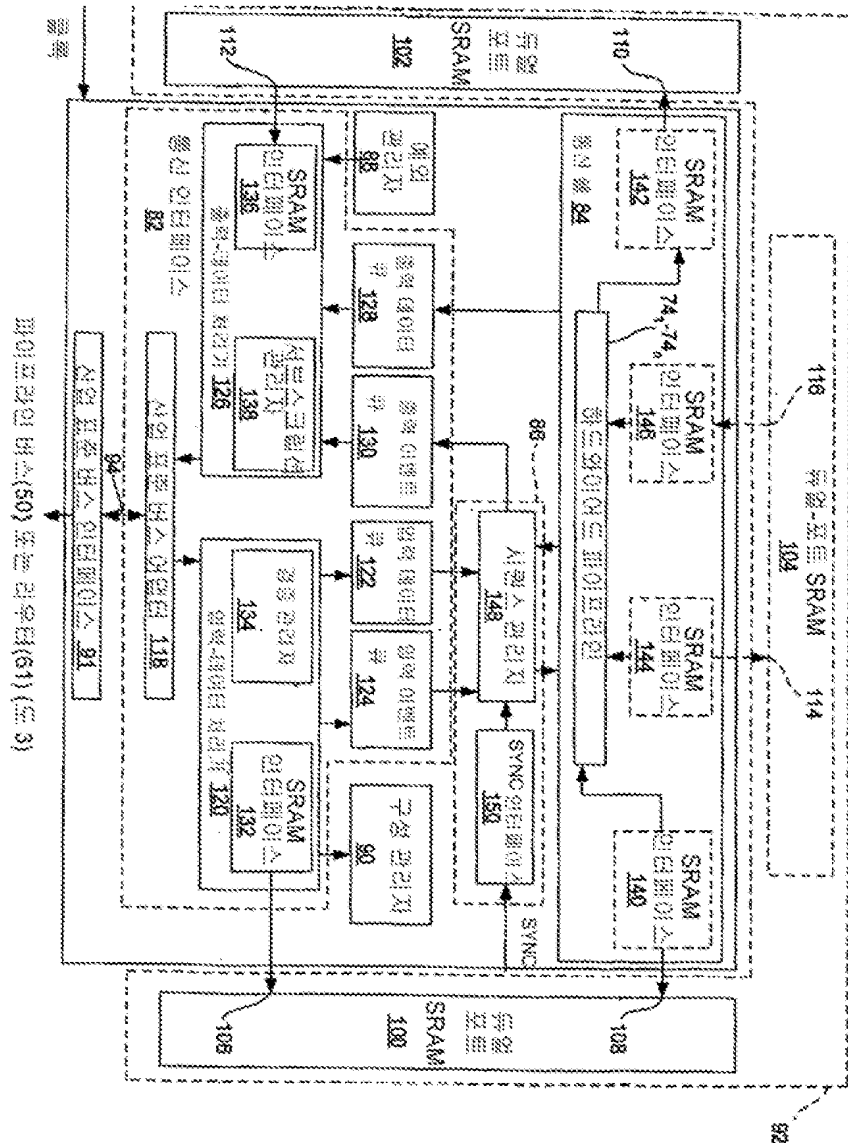


도면4

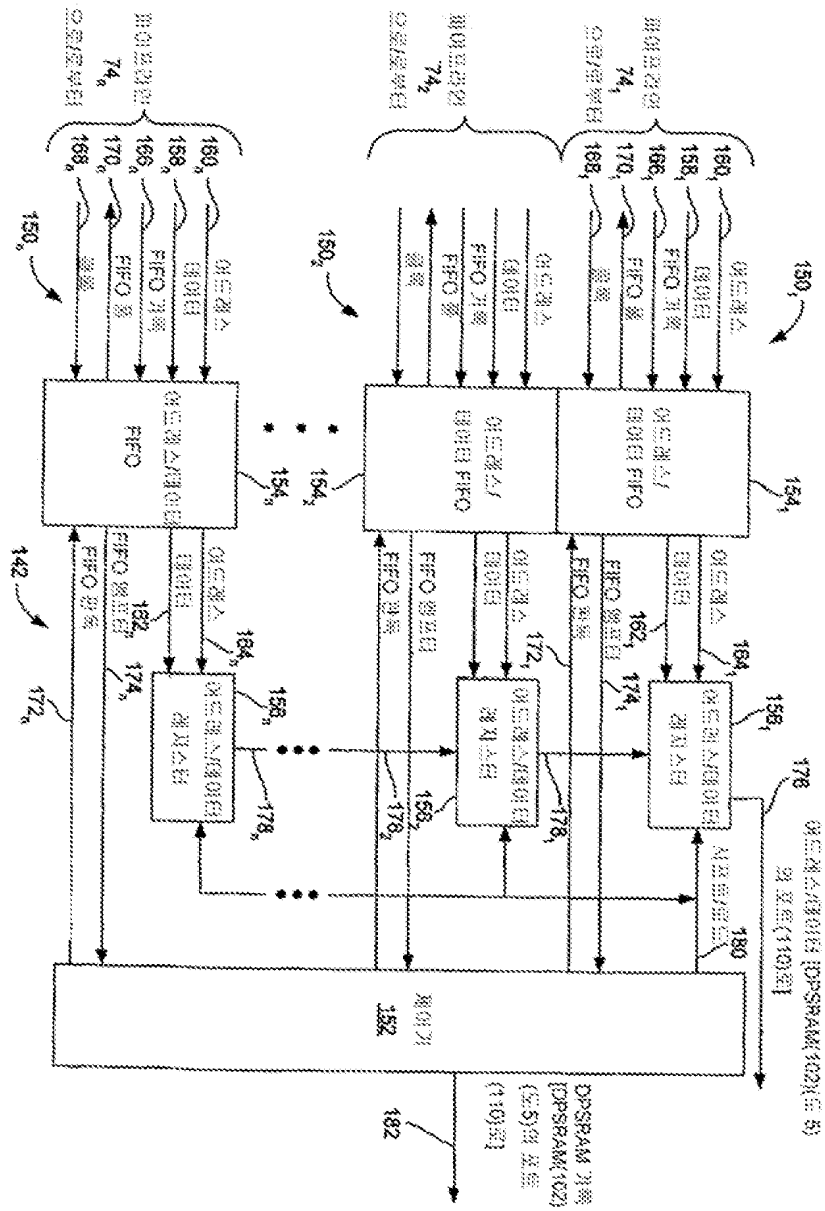




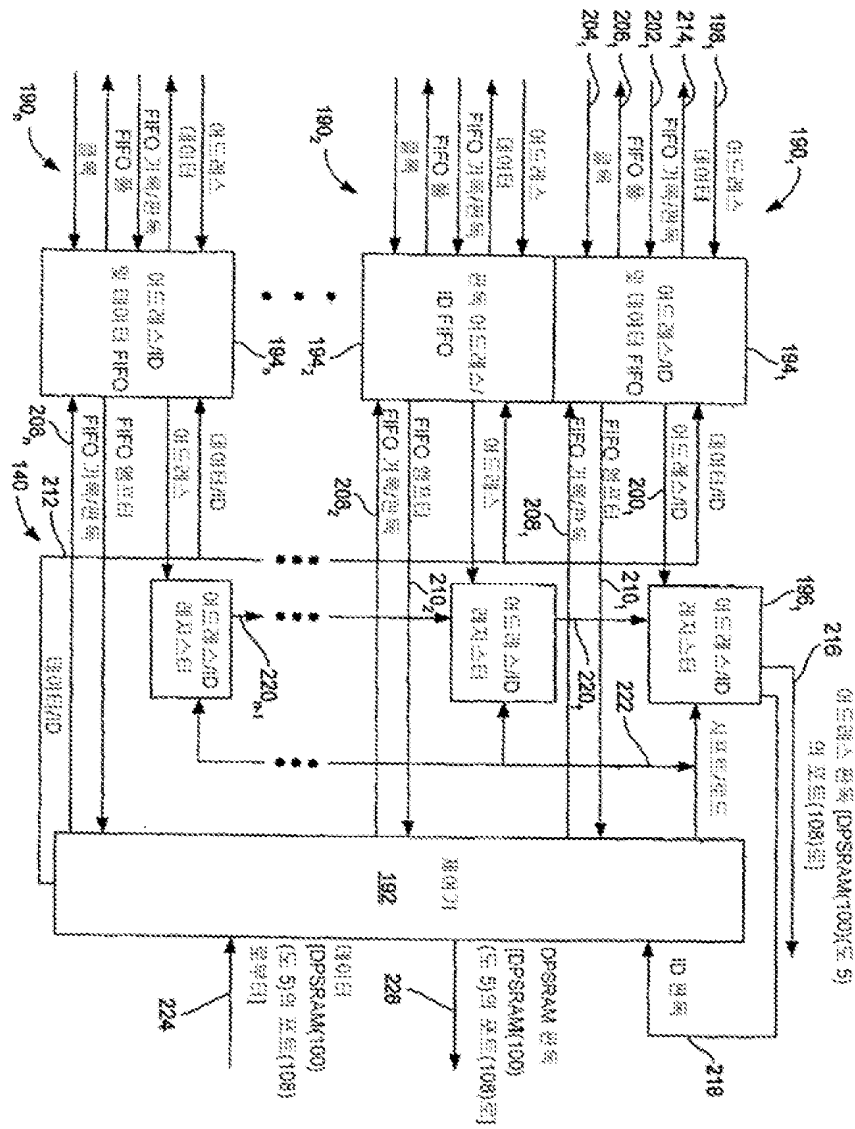
5936



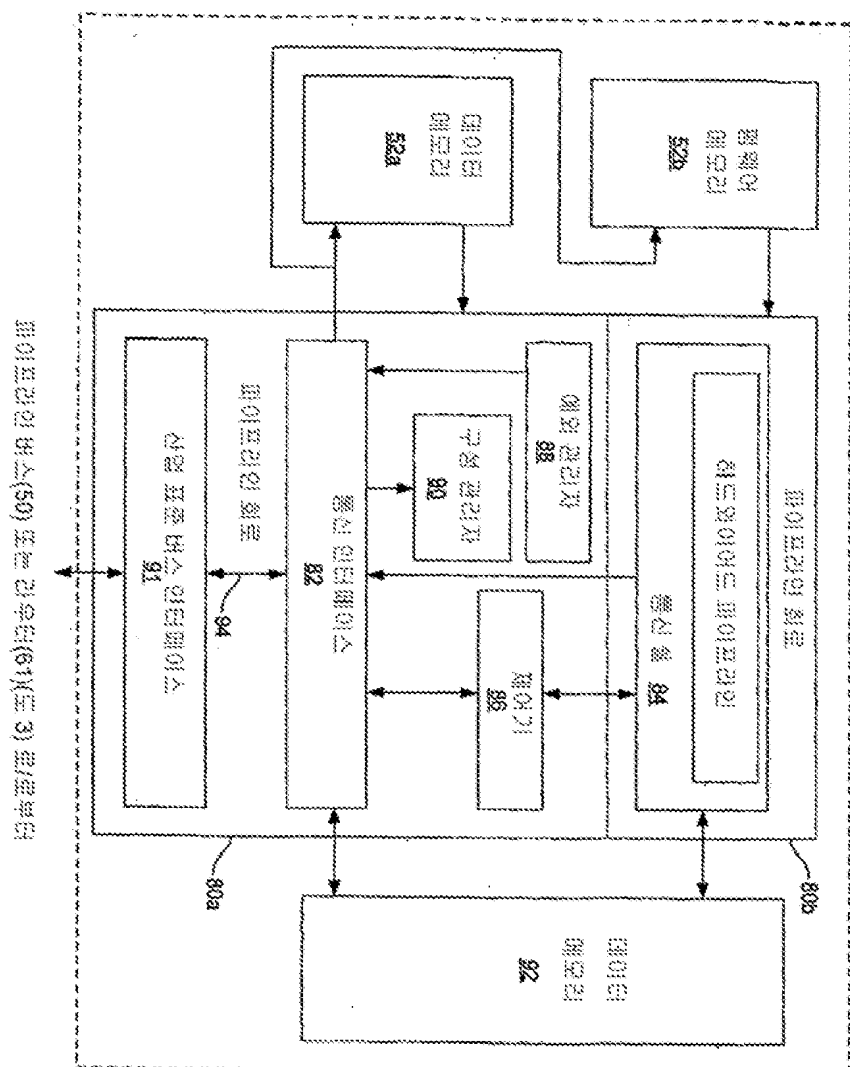
도 9



도면7



598



도면9

